

PROCEEDINGS:

IMAGE UNDERSTANDING WORKSHOP

OCTOBER 1984

Sponsored by:

Information Processing Techniques Office

Defense Advanced Research Projects Agency

IMAGE
SEQUENCE

DTIC FILE COPY

DEPTH
SEGMENTATION

AD-A149 496

DISPLACEMENT
VECTORS

This document has been approved
for public release and sale; its
distribution is unlimited.

DTIC
DEC 3 1984

84 10 23 025

SAIC
Science Applications International Corporation

12

IMAGE UNDERSTANDING

Proceedings of a Workshop
Held at
New Orleans, Louisiana
October 3-4, 1984

Sponsored by the
Defense Advanced Research Projects Agency

Science Applications International Corporation
Report Number SAIC-84/1726
Lee S. Baumann
Workshop Organizer and
Proceedings Editor

This report was supported by
The Defense Advanced Research
Projects Agency under DARPA
Order No. 3456, Contract No. MDA903-84-C-0160
Monitored by the
Defense Supply Service, Washington, D.C.

APPROVED FOR PUBLIC RELEASE
DISTRIBUTION LIMITED

DTIC

DEC 3 1984

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of the Defense Advanced Research Projects Agency or the United States Government.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SAIC-84/1726	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) IMAGE UNDERSTANDING Proceedings of a Workshop, October 1984		5. TYPE OF REPORT & PERIOD COVERED ANNUAL TECHNICAL June 1983 - October 1984
7. AUTHOR(s) LEE S. BAUMANN (Ed.)		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS SCIENCE APPLICATIONS INTERNATIONAL CORPORATION 1710 Goodridge Drive, 10th Floor McLean, Virginia 22102		8. CONTRACT OR GRANT NUMBER(s) MDA903-84-C-0160
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS ARPA ORDER No. 3456
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE October 1984
		13. NUMBER OF PAGES 343
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) APPROVED FOR RELEASE; DISTRIBUTION UNLIMITED		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Digital Image Processing; Image Understanding; Scene Analysis; Edge Detection; Image Segmentation; CCDArrays; CCD Processors		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This document contains the technical papers and outlines of semi-annual progress reports presented by the research activities in Image Under- standing, sponsored by the Information Processing Techniques Office; Defense Advanced Research Projects Agency. The papers were presented at a workshop conducted on 3-4 October 1984, in New Orleans, Louisiana.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

	<u>PAGE</u>
FORWARD	i
DEFENSE TECHNICAL INFORMATION CENTER ACCESSION NUMBERS	iii
AUTHOR INDEX	iv

SECTION I - PROGRAM REVIEWS BY PRINCIPAL INVESTIGATORS

"Image Understanding Research at the University of Massachusetts", E. M. Riseman and A. R. Hanson; University of Massachusetts	1
"Image Understanding Research at Columbia" J. R. Kender; Columbia University	6
"The SRI Image Understanding Research Program" M. A. Fischler; SRI International	8
"MIT Progress in Understanding Images" T. Poggio; Massachusetts Institute of Technology	14
"Spatial Understanding", T. O. Binford; Stanford University	25
"Recent Results of the Rochester Image Understanding Project", J. A. Feldman, D. H. Ballard, C. M. Brown; University of Rochester	29
"Image Understanding Research at USC: 1983-1984" R. Nevatia; University of Southern California	33
"Image Understanding Research at CMU", T. Kanade; Carnegie-Mellon University	42
"Image Understanding Techniques for Autonomous Vehicle Navigation", A. Rosenfeld, L.S. Davis A. M. Waxman; University of Maryland	48

SECTION II - TECHNICAL REPORTS PRESENTED

"Real-Time Image Processing at Westinghouse: 1964-1984", B. J. Schachter; Westinghouse Defense Center	53
"VLSI Implementation of Systolic and 3-D Cellular Architectures for Image Processing", J. G. Nash, R. D. Etchells, J. Grinberg, S. Hansen, M. J. Little, G. R. Nudd, K. Petrozolin, R. Turk; Hughes Research Laboratories	56

<div style="border: 1px solid black; border-radius: 50%; width: 40px; height: 40px; margin: 0 auto; display: flex; align-items: center; justify-content: center;"> <div style="writing-mode: vertical-rl; transform: rotate(180deg); font-size: 8px;">Date</div> <div style="font-size: 24px; font-weight: bold;">11</div> </div>	<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	STIS DTIC TAB Unannounced Justification	By	Distribution/ Availability Codes	Avail and/or Special	Dist <div style="font-size: 24px; font-weight: bold;">A-1</div>
---	---	--	----	-------------------------------------	-------------------------	--

TABLE OF CONTENTS

SECTION II - TECHNICAL REPORTS PRESENTED (Continued) PAGE

"Semantic Network Array Processor and Its Applications to Image Understanding", V. Dixit, D. I. Moldovan; University of Southern California.....	65
"Extended Gaussian Images", B. K. P. Horn; Massachusetts Institute of Technology	72
"Symmetry Evaluators", S. A. Friedberg, C. M. Brown; University of Rochester	90
"Graphics and Prediction from Models", R. Scott; Stanford University.....	98
"The Information-Centered Approach to Optimal Algorithms Applied to the 2-1/2D Sketch", J. R. Kender, D. Lee; Columbia University	107
"Determining 3-D Motion and Structure from Optical Flow Generated by Several Moving Objects", G. Adiv; University of Massachusetts	113
"Dynamic Stereo: Passive Ranging to Moving Objects from Relative Image Flows", A. M. Waxman S. S. Sinha; University of Maryland	130
"The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images", M. Herman, T. Kanade; Carnegie-Mellon University	137
"Hierarchical Warp Stereo" L. H. Quam; SRI International	149

SECTION III - TECHNICAL PAPERS NOT PRESENTED

"On Detecting Edges", V. S. Nalwa; Stanford University.....	157
"Extracting Straight Lines", J. B. Burns, A. R. Hanson, E. M. Riseman; University of Massachusetts	165
"Matching Closed Contours", K. E. Price, University of Southern California.....	169
"Edge Localization in Both Φ and X ", A. P. Blicher; Stanford University; University of California, Berkeley	176

TABLE OF CONTENTS

<u>SECTION III - TECHNICAL PAPERS NOT PRESENTED (Continued)</u>	<u>PAGE</u>
"Shading into Texture", A. P. Pentland; SRI International	179
"Interest Points, Disparities and Correspondence" A. Bandyopadhyay; University of Rochester	184
"A Theory of Line Drawing Interpretation" J. Malik, T. O. Binford; Stanford University	188
"Hierarchical Knowledge-Directed Object Extraction Using a Combined Region and Line Representation", G. Reynolds N. Irwin, A. Hanson, E. Riseman; University of Massachusetts	195
"Automatic Reformulation of Algorithms in Computational Geometry", M. Lowry; Stanford University	205
"A Fast Surface Interpolation Technique", G. B. Smith; SRI International	211
"The Hough Transform Method on Fine-Grained Tree-Structured SIMD Machines", H. A. H. Ibrahim, J. R. Kender, D. E. Shaw; Columbia University	216
"Controlled-Smoothness Stabilizers for the Regularization of Ill-Posed Visual Problems Involving Discontinuities" D. Terzopoulos; Massachusetts Institute of Technology	225
"Spatial Reasoning From Line Drawings of Polyhedra" T. M. Strat; SRI International	230
"Computing Dense Fields Displacement with Confidence Measures In Scenes Containing Occlusion", P. Anandan, University of Massachusetts	236
"Map-Guided Feature Extraction From Aerial Imagery" D. M. McKeown, Jr., J. L. Denlinger; Carnegie-Mellon University	247
"Ill-Posed Problems and Regularization Analysis in Early Vision", T. Poggio, V. Torre; Massachusetts Institute of Technology	257
"Recovering the Camera Parameters From a Transformation Matrix", T. M. Strat; SRI International	264

TABLE OF CONTENTS

<u>SECTION III - TECHNICALS PAPERS NOT PRESENTED</u>	<u>PAGE</u>
"Optical Navigation by the Method of Differences" B. D. Lucas, T. Kanade; Carnegie-Mellon University	272
"Autonomous Scene Description with Range Imagery; D. R. Smith, T. Kanade; Carnegie-Mellon University	282
"Description of 3-D Surfaces Using Curvature Properties", G. Medioni, R. Nevatia; University of Southern California	291
"The ASTERIX-System: A Feature Based Approach to the Correspondence Problem", L. S. Dreschler-Fischer, E. E. Triendl; Stanford University	300
"Stereo Modeling System: A Geometric Modeling System for Modeling Object Instance and Class" J. Takamura, T. O. Binford; Stanford University	302
"Acronym Model Based Vision in the Intelligent Task Automation Project", D. M. Chelberg, H. S. Lim, C. K. Cowan; Stanford University	308
"Iconic to Symbolic Processing Using a Content Addressable Array Parallel Processor", D. Lawton, S. Levitan, C. Weems, E. Riseman, A. Hanson; University of Massachusetts	316
"Survey of Array Processors", H. S. Lim, T. O. Binford; Stanford University.....	334

FOREWORD

On October 3-4, 1984, the Defense Advanced Research Projects Agency, Information Processing Techniques Office held a workshop on Image Understanding. This workshop was the fifteenth in the series which have been conducted by DARPA since 1975. The purpose of the workshop was to bring together the research community and the user community so that each could benefit from the interaction and, at the same time, promote a synergism which would improve the efforts of the individual participants as well as the communities as a whole. Toward this end, each Principal Investigator under the DARPA program reviewed progress during the past year and research personnel presented specific technical details of selected facets of their research work. This Proceedings incorporates the P.I. reports and the technical reports presented at the workshop and, in the interests of providing a total record of the research program, includes copies of those technical papers which were not presented due to the press of time.

Commander Ronald B. Ohlander, Assistant Director for Computer Science for the DARPA/IPTO, and the program manager for the Image Understanding research program chose as a theme for this year's workshop "Future Directions for I.U." The community should, he observed, reassess itself and take a close look at where it is headed in the next five years. Commander Ohlander believes that we should be concerned with new architectures for high-level vision processing, the use of techniques from other areas of AI, such as expert systems technology, and major thrusts in modeling and representation systems. With this in mind, a feature of this year's workshop was the inclusion of a special panel discussion session to air various views on the future directions for Image Understanding, and an overview by Commander Ohlander of the I.U. portions of the Strategic Computing Program, one of the major technology thrusts being undertaken by the Defense Advanced Research Projects Agency.

This proceedings has been supplied to the Defense Technical Information Center (DTIC) and copies may be secured from that Agency by writing to the following address:

Defense Technical Information Center
Cameron Station, Bldg. #5
Alexandria, VA 22314

A small charge is assessed by the DTIC for reproduction expenses. Accession number for this proceedings is not yet available but will be assigned by the DTIC within the next thirty days. Accession numbers for previous issues are listed on the following page.

The images chosen for the cover design of this proceedings were provided by the Computer and Information Science Department, Lederle Graduate Research Center, University of Massachusetts at Amherst. The images were essentially taken from Daryl Lawton's thesis* and are meant to illustrate the type of work that can be obtained from motion processing. The notes accompanying the images describes the process as follows:

The three photos shown on the left side represent successive frames of an image sequence taken from a vehicle moving along a road. The images show a road sign in the foreground, a telephone pole in the mid-ground, and a background of tree foliage.

In the center, the displacement vectors show the zero-crossing contours of the Laplacian of the images in gray. Superimposed on these, in white, are the locations of interest points along these contours. The lines extending from these points show the displacement of these interest points through subsequent frames along radial flow lines emanating from the focus of expansion of the optic flow, which is found by an optimizing search procedure.

Once the focus of expansion has been determined, displacement vectors can be computed along almost all contours, even those on which there are no interest points. From these displacements, the environmental depth of the contours can be calculated. For the depth segmentation, as shown in the three right hand photos, the contours are separated into intervals of environmental depth. This fairly well extracts the road sign, the telephone pole, and the background foliage, which lie at distinct ranges of increasing depth from the observer.

As usual, the artwork and lay out for the Proceedings cover is the work of Mr. Tom Dickerson of Science Applications International Corporation. Appreciation is also due to Ms. Lori Beth DeFuria who handled all the mailings and Ms. Barbara Burkett and Ms. Barbara Ashooh who contributed typing support in putting together the proceedings.

Lee S. Baumann
Science Applications
International Corporation
Workshop Organizer

* Daryl T. Lawton, "Processing Dynamic Image Sequences from a Moving Sensor," Phd. Dissertation and Coins Technical Report 84-05, University of Massachusetts at Amherst, February 1984.

* Daryl T. Lawton, "Processing Restricted Sensor Motion", Proceedings: Image Understanding Workshop, June 1983, Pps 266-281.

DEFENSE TECHNICAL INFORMATION CENTER
ACCESSION NUMBERS FOR PREVIOUS
I.U. WORKSHOPS

<u>ISSUE</u>	<u>DATE</u>	<u>NUMBER</u>
APRIL	1977	ADA 052900
OCTOBER	1977	ADA 052901
MAY	1978	ADA 052902
NOVEMBER	1978	ADA 064765
APRIL	1979	ADA 069515
NOVEMBER	1979	ADA 077568
APRIL	1980	ADA 084764
APRIL	1981	ADA 098261
SEPTEMBER	1982	ADA 120072
JUNE	1983	ADA 130251

AUTHOR INDEX

<u>NAME</u>	<u>PAGE</u>
Adiv, G.	113
Anandan, P.	236
Ballard, D.H.	29
Bandyopadhyay, A.	184
Binford, T.O.	25, 188, 302, 334
Blicher, A.P.	176
Brown, C.M.	29, 90
Burns, J.B.	165
Chelberg, D.M.	308
Cowan, C.K.	308
Davis, L.S.	48
Denlinger, J.L.	247
Dixit, V.	65
Dreschler-Fischer, L.S.	300
Etchells, R.D.	56
Feldman, J.A.	29
Fischler, M.A.	8
Friedberg, S.A.	90
Grinberg, J.	56
Hansen, S.	56

AUTHOR INDEX (Cont'd)

<u>NAME</u>	<u>PAGE</u>
Hanson, A.R.	1, 165, 195, 316
Herman, M.	137
Horn, B.K.P.	72
Ibrahim, H.A.H.	216
Irwin, N.	195
Kanade, T.	42, 137, 272, 282
Kender, J.R.	6, 107, 216
Lawton, D.	316
Lee, D.	107
Levitan, S.	316
Lim, H.S.	308, 334
Little, M.J.	56
Lowry, M.	205
Lucas, B.D.	272
Malik, J.	188
McKeown, D.M., Jr.	247
Medioni, G.	291
Moldovan, D.I.	65
Nalwa, V.S.	157
Nash, J.G.	56
Nevatia, R.	33, 291

AUTHOR INDEX (Cont'd)

<u>NAME</u>	<u>PAGE</u>
Nudd, G.R.	56
Pentland, A.P.	179
Petrozolin, K.	56
Poggio, T.	14, 257
Price, K.E.	169
Quam, L.H.	149
Reynolds, G.	195
Riseman, E.M.	1, 165, 195, 316
Rosenfeld, A.	48
Schachter, B.J.	53
Scott, R.	98
Shaw, D.E.	216
Sinha, S.S.	130
Smith, D.R.	282
Smith, G.B.	211
Strat, T.M.	230, 264
Takamura, J.	302
Terzopoulos, D.	225
Torre, V.	257
Triendl, E.E.	300
Turk, R.	56
Waxman, A.M.	48, 130
Weems, C.	316

SECTION I

PROGRAM REVIEWS

BY

PRINCIPAL INVESTIGATORS

**IMAGE UNDERSTANDING RESEARCH
AT THE UNIVERSITY OF MASSACHUSETTS**

Edward M. Riseman and Allen R. Hanson

Computer and Information Science Department
University of Massachusetts
Amherst, MA 01003

ABSTRACT

Our DARPA funded research program continues to focus on dynamic image processing. The work includes the hierarchical computation of more accurate displacement fields in the presence of occlusion, the computation of general motion parameters of the sensor and independently moving objects, and architectures which will allow real-time image interpretation. Closely related work involves a new low-level algorithm for extracting straight lines, even those with very low contrast, from natural scenes, and the development of a knowledge-based system for photointerpretation of aerial images.

**1. MOTION PROCESSING FOR
RECOVERY OF ENVIRONMENTAL DEPTH**

We continue to explore and evolve several classes of motion algorithms for processing image sequences from a sensor moving through an environment. The major goal in motion processing is the recovery of the motion parameters of the sensor and each independently moving object. The computation of environmental depth of visible surfaces follows in a rather straightforward manner.

In the first group, we have already reported Lawton's research [LAW83, LAW84] on a class of algorithms for restricted cases of sensor motion through a static environment; of particular interest are pure translational processing and motion restricted to a known plane. This algorithm combines the computation of the image displacements forming the flow field with the computation of the sensor motion parameters. It avoids many of the errors produced by noise, ambiguity, and occlusion when, as usually is the case, the flow field is computed prior to inference of the motion parameters. Current work on these algorithms is directed towards a more careful analysis of the accuracy, efficiency, and robustness of this approach. The performance of the

various algorithms is being quantified on a larger set of images. We intend to study the sensitivity of the algorithm to the number of feature points employed, the amount of texture exhibited in the scene, the orientation of the sensor relative to the direction of motion, the rate of convergence to a solution, and the accuracy of the final depth map.

A second major approach to motion, involving the analysis of general sensor motion in an environment containing independently moving objects, will be documented in the forthcoming thesis of Adiv [ADI84a]. This motion algorithm first approximates the visual field as rigid motion of a set of planar surfaces, groups them into rigidly moving objects, and then drops the planarity assumption to infer the 3D motion and depth of each of these segments. Let us consider the issues in a little more detail.

The most common approach for the analysis of visual motion is based on two phases: computation of an optical flow field and interpretation of this field. A major problem which has emerged in this research area is sensitivity to noise. Flow fields generated by existing techniques are noisy and partially incorrect, especially near occlusion or motion boundaries. Most of the algorithms for interpreting these fields cannot successfully deal with realistic levels of noise. Global approaches, which utilise all the available information, can be expected to be relatively robust. Still, an inadequate choice of an optimization criterion often limits the performance of these techniques. Furthermore, the presence of independently moving objects usually makes such global techniques impractical.

These two issues, the presence of noise and the presence of independently moving objects, are addressed by a new scheme for interpreting optical flow fields, which are allowed to be either dense or sparse [ADI84b]. In the first stage of this scheme the flow field is segmented into connected sets of flow vectors, where each set is consistent with a rigid motion of a roughly planar surface and, therefore, is likely to be associated with only one rigid object. The algorithm for achieving such a segmentation is based on a modification of the generalised Hough technique, in which flow vectors are grouped into components consistent with affine transformations of the image. If appropriate, components are then merged to create segments.

¹ This work has been supported primarily by the Defense Advanced Research Projects Agency under Contract N00014-82-K-0464. Some of the work reported here has been supported by the AFOSR under Contract F49620-83-C-0099 and by RADC under Task I-4-0055.

In the second stage of the proposed scheme sets of segments are hypothesized to be induced by the same rigidly moving object. Each of these hypotheses is tested by searching for 3-D motion parameters which are compatible with all the segments in the corresponding set. This search is based on a least-squares technique which minimises the deviation between the given flow field and that predicted from the computed parameters. Once the motion parameters are recovered, the relative environmental depth can be estimated.

This technique, of segmenting the flow field and, then combining segments to form object hypotheses, makes it possible to deal with independently moving objects while employing all the available information associated with each object. In addition, the optimisation criterion for recovering 3-D information is appropriate for dealing with partially incorrect flow fields. Thus, the proposed scheme is relatively insensitive to noise. Experiments, based on real and simulated data, involve flow fields which are noisy and partially incorrect, especially near occlusion boundaries [ADI84b]. The successful results demonstrate the effectiveness of the scheme in such situations. There are, however, inherent ambiguities in the interpretation of noisy flow field. These ambiguities will be analysed and demonstrated in [ADI84a].

2. COMPUTING ACCURATE DISPLACEMENT FIELDS IN THE PRESENCE OF OCCLUSION

One method for computing displacement fields prior to depth and motion parameter computations is by area correlation. However, the displacement fields produced by this method are often incorrect in homogeneous areas of the image and near occlusion boundaries (where an area is visible in one frame and occluded in the next). Anandan [ANA84a,b] has developed a matching algorithm which overcomes many of the problems with current techniques. The algorithm is obtained by modifying the search strategy employed in the hierarchical algorithm of Glaser, Reynolds, and Anandan [GLA83] to take into account the reliability of each displacement vector as provided by a confidence measure. The result is a computationally efficient matching algorithm which provides a dense displacement field with estimates of the reliability of each displacement vector.

The matching algorithm developed by Glaser et al is a hierarchical coarse-fine strategy using band-pass filtered images in which matches found at the coarse (low-frequency) levels direct the search for matches at the finer (higher frequency) levels. In this algorithm, each pixel at a coarse level covers a large area at the finest level; errors at the coarse level cause incorrect initial estimates to be generated for the finer levels and hence the search at the finer levels is carried out in areas which do not include the correct match. The confidence measure attempts to detect these and similar situations so that the search strategy can be modified accordingly.

The confidence measure used is based on the shape of the SSD surface generated by the values of the sum-of-squared-differences corresponding to different candidate

displacements from the pixel. For example, the SSD surface tends to have a rather flat behavior in homogeneous areas of the image as opposed to a much sharper valley in areas where there is distinct image structure. The confidence measure is computed as the minimum of the normalised second derivatives of the SSD surface computed at 0, 45, 90, and 135 degrees by a 1×3 Laplacian operator centered at the point of best match. This measure tends to be low in homogeneous and occluded areas, and along edges; hence it is useful in isolating the areas in an image where the displacement estimates are unreliable and often incorrect.

The original search strategy described in Glaser started at a coarse level at which all image displacements were at most one pixel. Matches found in a 3×3 window were projected to the four pixels at the next finer level of resolution and a search conducted in a 3×3 window around these estimates. This process continued down to the finest level of resolution. Anandan modifies this search strategy in two ways:

1. Only high confidence coarse estimates are projected. The motivation for this is that when incorrect coarse estimates are projected, the 3×3 searches at the finer levels are conducted in areas of the second frame that do not include the true match point, which in turn causes incorrect searches at all subsequent levels.
- and 2. Estimates may be projected to an area larger than the four descendant pixels. If the incorrect coarse level projections are eliminated, then the finer level search can be conducted over a larger area and the true match can perhaps be recovered.

The modified algorithm appears to work well in complex image pairs. Future work involves using the confidence measure and other attributes of the SSD surface to distinguish between occluded and homogeneous areas and to use directional information in the SSD surface to modify the expansion of the search area in a direction perpendicular to an edge.

3. EXTRACTING STRAIGHT LINES

The organisation of local intensity changes into the more global abstractions called "lines" or "boundaries" is an early and important step in the transformation of the visual signal into intermediate constructs useful for interpretation processes. A recent algorithm developed by Burns [BUR84a,b,c] uses gradient orientation, rather than changes in image intensity, as the initial organising feature. The general approach consists of four basic steps:

- 1) Group pixels into line-support regions based on similarity of gradient orientation. This allows data directed organisation of edge contexts without commitment to masks of a particular size.
- 2) Approximate the intensity surface by a planar surface. The planar fit is weighted by the gradient magnitude associated with the pixels so that intensities in the steepest part of the edge will dominate.

- 3) **Extract attributes** from the line-support region and the planar fit. The attributes extracted include the representative line and its length, contrast, width, location, orientation, and straightness.
- 4) **Filter lines** on the attributes to isolate various image events such as long straight lines of any contrast; high contrast short lines (heavy texture); low contrast short lines (light texture); homogeneous regions of adjacent very low contrast lines; and lines at particular orientations and positions.

Gradient orientation at a pixel is estimated from the horizontal and vertical components of the gradient obtained from two 2×2 masks. These estimates are then grouped into regions by using two overlapping sets of partitions of fixed size, say two 45° sets staggered by 22.5° or two 22.5° sets staggered by 11.25° . Each set produces a region segmentation of the gradient image and then these segmentations are merged by choosing that region satisfying local constraints. Each resulting "line support region" is a candidate area for a straight line since the local gradient estimates share a common orientation. A representative line can be extracted by intersecting a least squares planar estimate of the underlying intensity surface with the plane corresponding to the average region intensity.

A set of attributes is extracted from the line support region and its associated line. These attributes include line length, contrast, width, steepness, straightness (or deviation from straightness), and various orientation and position parameters. They form the basis of a line data base for an image over which various filtering operations can be performed in order to extract lines with specific properties. For example, in many of our images long high contrast lines correspond to significant boundaries, medium contrast short lines may correspond to textured areas, and low contrast wide lines to slow intensity gradients.

The algorithm does an excellent job of extracting straight lines, including very low contrast lines. It represents all lines as straight lines and we are examining ways in which the same general technique could be used to generate curvilinear lines and boundaries. The gradient orientation grouping operator does a credible job for such a simple technique and we are looking at methods for detecting and overcoming the overgrouping and other errors it sometimes produces. Finally, we are examining a variety of line descriptors and are incorporating them into other interpretation processes (e.g., see [REY84b] in this proceedings).

4. INTERPRETATION OF AERIAL PHOTOGRAPHS

The detailed examination, segmentation, and understanding of high resolution digital images represents a severe computational load for current computers. One technique for reducing the overall computational requirements involves selectively focussing on relevant portions of an image and ignoring irrelevant portions. The specification of relevancy implies some external model which represents a

description of those areas or objects that are of potential interest and to which computational resources may most fruitfully be applied. The most suitable method for applying such selective processing to high resolution imagery is the multi-resolution, or pyramid, technique. From the original, large-scale, full resolution image is constructed a progression of smaller and smaller images, each covering the same extent, but at successively coarser resolution.

In this section we describe some recent experiments using a hierarchical segmentation algorithm and focus of attention mechanism for locating buildings, roads, and airports in a high-resolution monochromatic aerial image. The approach involves formulating segmentation and feature extraction algorithms as hierarchical algorithms within the processing cone [HAN80]. The focus of this section is on the segmentation processes; more complete interpretation results may be found in [REY84a,b].

The general idea is to use the Nagin-Kohler localized histogram region segmentation algorithm [NAG79, KOH83] and the Burns straight line extraction algorithm as the primary low-level processes used to drive the bottom-up component of a hierarchical localized segmentation process. The feature extraction process yields a low-level representation of the data and an evidential-based inference net is used to transform this data to an intermediate level of representation within long-term memory. This intermediate level of representation in turn allows the multiresolution segmentation algorithms to be focussed and selectively applied to areas of interest in the image. We are investigating the effectiveness of directing the system to look only in areas where a coarse level segmentation yields a hypothesis that an object of the sort we are looking for exists. For high-resolution imagery, the computational advantage of this approach is significant. For example if we assume that even $2/3$ of the possible sectors are used at each level, then only $1/5$ of the image is being examined 4 levels down. In the case that only $1/4$ of the sectors are selected, only $1/256$ th of the image is being searched 4 levels down. Thus the computational complexity of the process can be kept within reasonable bounds.

The selection of candidate regions for examination at a higher resolution is accomplished by choosing all regions which satisfy a set of object dependent constraints on region and line attributes. In general, the results from such a simple rule will be unreliable and prone to error. Image interpretation is implicitly involved with the problems associated with combining information from multiple sources of knowledge. Any perceptual system which utilises processed sensory data must recognise the fact that to varying degrees the information will be imperfect and prone to error. With this in mind we are developing mechanisms for evidential reasoning and inferencing under uncertainty [LOW82, WES82] in order to construct more reliable focussing sets.

Some of the limitations of inferencing using Bayesian

probability models are overcome using the Dempster-Shafer formalism for evidential reasoning, in which an explicit representation of partial ignorance is provided [SHA76]. The inferencing model allows "belief" or "confidence" in a proposition to be represented as a range within the [0,1] interval. The lower and upper bounds represent support and plausibility, respectively, of a proposition, while the width of the interval can be interpreted as ignorance. Wesley [WES83] is extending this approach to the problem of distributed control of a set of knowledge sources which can be applied to examine particular concepts in long-term memory.

Once the inference network is fully integrated, we expect the hierarchical segmentation and interpretation process to operate as follows. First the local histogram-based region segmentation and the linear feature extraction algorithm are applied at a coarse level of resolution. Knowledge sources in the form of object hypothesis rules are then applied to region and line attributes at that level. The output of these rules is converted to a form appropriate for input into the inference network of long-term memory. The inferencing process is then invoked and each region yields a support and plausibility (i.e., a range of belief) that it is a candidate region for one of the goal objects. The region and line segmentation algorithms are then applied at a finer level of resolution, but only on the candidate regions which have high support. At this finer level of resolution the representation of the object is of a different form and may involve more expensive object rule combinations of the region and line attributes, but applied only to a small subset of the image. The process is recursively applied to finer levels of resolution.

5. CONTENT ADDRESSABLE ARRAY PARALLEL PROCESSOR (CAAPP)

For the last several years there has been a synergistic relationship between our machine vision group and a parallel architecture group led by Professor Caxton Foster [WEE84, LEV84]. We have designed and are now proposing the construction of a large scale Content Addressable Array Parallel Processor (CAAPP) for low, medium and high level vision processing. This new architecture combines associative processing via global broadcast and response to and from an array of cells, and array processing via local cellular square neighborhood computation. The resulting architecture allows simple solutions to many problems that are difficult for parallel machines which provide only one of these capabilities [WEE83].

The prototype CAAPP would consist of sixteen thousand processing elements arranged as a 128×128 square array. We have taken a pragmatic view of VLSI technology because we wish to actually construct this machine. Hence, we have approached the design in a very conservative manner, making use only of existing technology (3μ NMOS), to insure rapid and successful development. This design is intended to be expandable up to at least a quarter of a million processing elements in a 512×512 array. The initial

16K processor parallel machine will have an effective operating speed several hundred to a thousand times that of the fastest sequential processor available today. The CAAPP would be connected via its own controller to a VAX-11/780, LISP machine, or some other general purpose computing machine which would provide both the algorithm development environment and the operating environment for the system.

The CAAPP is capable of providing an intermediate symbolic representation by storing the results of low level vision algorithms and providing the communication interface to knowledge-based processing [FOS84]. In addition to its two dimensional communication pathways between neighboring cells and bit-serial local cellular computation, this device would be capable of content addressable memory (associative) functions including broadcast instruction, find-first-responder, and count-responders. These operations permit the feedback loop to be closed between high-level processing and low level processing by allowing communication and control information to flow up and down between the different representations supported by the architecture.

The intermediate level of representation provides an interface between the low and high levels of representation, that is, between pixel-based representation and symbolic elements representing visual knowledge stored in a database. In the UMASS VISIONS system, the intermediate level consists of a symbolic description of the two dimensional image in terms of regions and line segments (that are still in registration with the raw image data) as well as their associated attributes which can be used in the interpretation process. In some systems this level would consist of representations of surfaces, or more generally, "intrinsic" features of the physical environment.

Intermediate processing includes several kinds of activities. First is the set of bottom-up tasks which are needed to complete the intermediate level of representation. This includes the extraction of the features for regions, lines, and vertices as well as the relations between these entities. The second group of intermediate processing activities involve grouping, splitting, and labelling processes, in either data-directed or knowledge directed modes (i.e., bottom-up or top-down) to form intermediate events which more naturally match stored object descriptions.

We believe that *the key to vision processing is a flow of communication and control both up and down through all representation levels*. Communication between these levels is by no means unidirectional. In most cases, recognition of an object or part of a scene at the high level will establish a strategy for further processing and probing the low and intermediate levels, in order to pull out additional features under the guidance of a partial interpretation. This might involve joining together region, line, and surface information to form a symbolic representation would more easily and naturally match a stored object description.

REFERENCES

- [ADI84a] Adiv, G., *The Interpretation of Optical Flow*, Ph.D. Thesis, Computer and Information Science Department, University of Massachusetts at Amherst, in preparation (1984).
- [ADI84b] Adiv, G., *Determining 3-D Motion and Structure from Optical Flow Generated by Several Moving Objects*, (these proceedings).
- [ANA84a] Anandan, P., *Computing Dense Displacement Fields with Confidence Measures in Scenes Containing Occlusion*, COINS Technical Report ((forthcoming)), University of Massachusetts at Amherst.
- [ANA84b] Anandan, P., *Computing Dense Displacement Fields with Confidence Measures in Scenes Containing Occlusion*, (these proceedings).
- [BUR84a] Burns, J.B., Hanson, A.R. and Riseman, E.M., *Extracting Linear Features*, Proc. of the 7th International Conference on Pattern Recognition (July 30 - August 2, 1984), Montreal, Canada.
- [BUR84b] Burns, J.B., Hanson, A.R. and Riseman, E.M., *Extracting Straight Lines*, COINS Technical Report (September 1984), University of Massachusetts at Amherst.
- [BUR84c] Burns, J.B., Hanson, A.R. and Riseman, E.M., *Extracting Straight Lines*, (these proceedings).
- [FOS84] Foster, C., Riseman, E., Weems, C., et al, *Content Addressable Array Parallel Processor CAAPP*, Proc. of Workshop on Algorithm-Guided Parallel Architectures for Automatic Target Recognition (July 1984), Leesburg, VA.
- [GLA83] Glaser, F., Reynolds, G. and Anandan, P., *Scene Matching by Hierarchical Correlation*, Proc. of CVPR, Washington, D.C. (1983).
- [HAN80] Hanson, A. and Riseman, E., *Processing Cones: A Computational Structure of Image Analysis*, in "Structured Computer Vision", S. Tanimoto, ed., Academic Press, New York, 1980.
- [KOH83] Kohler, R., *Integrating Non-Semantic Knowledge into Image Segmentation Processes*, Ph.D. Dissertation and COINS Technical Report 84-04 (September 1983), University of Massachusetts at Amherst.
- [LAW83] Lawton, D., *Processing Restricted Motion*, Proc. of the DARPA Image Understanding Workshop (June 1983), Arlington, VA.
- [LAW84] Lawton, D., *Processing Dynamic Image Sequences from a Moving Sensor*, Ph.D. Dissertation and COINS Technical Report 84-05 (February 1984), University of Massachusetts at Amherst.
- [LEV84] Levitan, S., *Parallel Algorithms and Architectures: A Programmer's Perspective*, Ph.D. Dissertation and COINS Technical Report 84-11 (May 1984), University of Massachusetts at Amherst.
- [LOW82] Lowrance, J., *Dependency-Graph Models of Evidential Support*, Ph.D. Dissertation and COINS Technical Report 82-26 (September 1982), University of Massachusetts at Amherst.
- [NAG79] Nagin, P., *Studies in Image Segmentation Algorithms Based on Histogram Clustering and Relaxation*, Ph.D. Dissertation and COINS Technical Report 79-15 (September 1979), University of Massachusetts.
- [REY84a] Reynolds, G., Kitchen, L., et al, *Experiments in Automatic Feature Extraction - A Report to RADC*.
- [REY84b] Reynolds, G., Irwin, N., Hanson, A., et al, *Hierarchical Knowledge-Directed Object Extraction Using a Combined Region and Line Representation*, (these proceedings).
- [SHA76] Shafer, G., "A Mathematical Theory of Evidence", Princeton University Press, 1976.
- [WEE83] Weems, C., Levitan, S., Foster, C., et al, *Titanic Information Kit*, COINS Technical Report 83-32 (1984), University of Massachusetts at Amherst.
- [WEE84] Weems, C., *Image Processing with a Content Addressable Array Parallel Processor*, Ph.D. Dissertation and COINS Technical Report 84-14 (September 1984), University of Massachusetts at Amherst.
- [WES82] Wesley, L. and Hanson, A., *The Use of an Evidential-Based Model for Representing Knowledge and Reasoning about Images in the VISIONS System*, Proc. of the Workshop on Computer Vision (August 1982), 14-25, Rindge, New Hampshire.
- [WES83] Wesley, L., *Reasoning About Control: An Evidential Approach*, SRI Technical Note 324 (June 1984).

Image Understanding Research at Columbia

John R. Kender

Department of Computer Science
Columbia University, New York, NY 10027

1. Abstract

The Image Understanding Project at Columbia continues to center its efforts on basic "middle-level" vision research: the representations and algorithms concerned with deriving surface information from low-level aggregate cues. The project has expanded somewhat to cover six major concerns in image understanding: the complexity of algorithms, the theory and analysis of texture, the integration of systems, the development of research aids, and the exploitation of two parallel architectures. This report on our second full year summarizes our progress in each of these areas; we note the graduation of our first doctoral student.

2. Introduction

The Image Understanding Project at Columbia continues a steady growth. We have augmented our hardware and software base (Vax plus Grinnell, running melded CMU and SRI vision libraries) by incorporating an image processor board and by implementing a flexible convolution package. Our research is a bit wider in scope, although it still emphasizes that level of image understanding which moderates low-level cues into surface information; we have six investigations.

We have taken the information-centered approach to optimal algorithms and applied it to the sparse depth data interpolation task (and others), with surprising results. We are extending the shape-from-texture theory by investigating the applicability of scale-space texture filtering and fractal texture descriptions. Work continues on the integration of multiple texture algorithms into a coherent shape analysis system. We have pursued the development of various graphic aids for the image understanding experimenter. For the Non-Von supercomputer, we have simulated and critiqued a wide range of image understanding algorithms. For the Grinnell image processor, we have written and experimentally verified routines for highly efficient low-level edge detection.

3. The Complexity of Image Algorithms

Under independent development at Columbia is a theory of computational complexity called the information-centered approach to optimal algorithms. It has proven surprisingly successful in the analysis of image understanding tasks. We (David Lee) have applied it to the problem of reconstructing a surface from sparse depth information, and we have found that the problem is optimally solved, in constant time, by spline interpolation; no adaptive algorithm would perform better [8]. Since splines are easily computed by parallel SIMD algorithms, we foresee the possibility of very simple interpolation machines, based on binocularity or range finding. We (Terry Boult) have empirically investigated the computational properties of the splines on synthetic data, and discovered encouraging regularities in its behavior which we hope to exploit for high efficiency.

Preliminary results indicate that similar mathematical approaches to the problems of shape-from-shading and of optical flow may result in similar optimal algorithms. Thus, it may be possible to put a fair amount of low-level image understanding algorithms on a unified theoretical footing.

4. The Analysis of Texture

Many of the algorithms we have devised for our middle-level work are derived from a central methodological paradigm called "shape from texture" [9]. We have most recently applied the paradigm in two additional areas, deriving further surface constraint relations and procedures. We have shown how to exploit the gravitationally induced environmental labels such as "horizontal" and "vertical" [6], and how to exploit the assumptions of equality of linear extents such as equal spacing or widths [7].

In work done jointly with IBM, we (Paul Douglas) have begun to extend the theory by examining the textural residue that remains after a surface has been represented in its scale-space filtered form. We are pursuing several conjectures about the fractal dimension of this residual signal, relative to the surface's perceived orientation and distance. In related work, we have explored the use of simulated annealing as a control structure for the perception of "emergent" visual phenomena such as subjective groupings of textured surfaces.

5. Integrated Image Systems

Our (Mark Moerdler) work continues on a middle-level vision system that integrates knowledge about surfaces from multiple independent texture-based sources. Two knowledge sources have been refined, based on texture density and texture spacing cues; they are tested for accuracy on several synthetic (but noisy) images. The sources are beginning to develop specialized heuristics: for textural segmentation, and for the handling of degeneracies. Several other sources, based on texture orientation and textural area, have been implemented and are waiting to be incorporated into the system.

6. Image Research Aids

Image understanding systems produce vast amounts of complex intermediate data: the middle levels of vision generate multiple assertions about any underlying surface. We have explored some graphic aids to the perception of these hypotheses on surface orientation, but without great success. The use of sequins (circles seen in perspective so that their ellipsoidal shapes are suggestive of underlying slant and tilt) appears fairly promising, however.

In a related effort, we (David Freudenstein) are in the beginnings of the construction of an environment for the easy intermingling of images and textual information: for communication, documentation, and publication purposes.

7. Image Understanding on Non-Von

Under independent development at Columbia is the Non-Von supercomputer, one of a class of fine-grained tree-structured machines built of custom VLSI chips. We (Hussein Ibrahim, in his thesis [2]) have found that such architectures lend themselves easily and naturally to many low-level vision algorithms, and, with some care, to middle-level vision algorithms as well.

Vision algorithms were selected to span different levels of computer vision tasks [3]. They included image correlation, histogramming, connected component labeling [4], geometric property computations, set operations, the Hough transform method for detecting object boundaries [5], and the correspondence methods for moving light displays. The encoding of the algorithms incorporated novel approaches to reduce the effects of the communication bottleneck usually associated with tree architectures.

Performance was studied using two simulators: all algorithms were simulated on a functional level simulator, and some were also simulated on a machine instruction level simulator. In general, we have found the performance of this class of tree machines to be superior to other highly parallel architectures for image understanding problems. We have also identified the limitations of these architectures, and have proposed methods to ameliorate them.

In other related work done jointly with Bell Laboratories, we (Marcia Derr) have explored the nearly opposite problem of distributing visual computation over a very coarse-grained multiprocessor: several Motorola 68000s on a high-speed bus. No straightforward way of partitioning the algorithms or the images has proven fully effective over the entire class of problems considered.

8. Exploiting the Grinnell Image Processor

We have upgraded our Grinnell with an Image Processor board, whose primitive 16-bit ALU imbedded in the memory refresh-cycle performs as an extremely fine-grained SIMD machine. After writing a general purpose convolution package for it [1], we (Christian Fortuvel and John Kender) were able to design for it very fast parallel algorithms to do edge detection. By various stratagems we have reduced the $O(\sigma^2)$ time for a series of Laplacian of Gaussian to $O(1)$, with very little loss of accuracy.

Briefly, the two-dimensional band-pass operators can be decomposed into sums or differences of other functions in multiple ways; these component functions can further be separated by variables into outer products of one-dimensional functions. Intermediate results can be calculated so that partial computations are cascaded. Further time reductions are possible by expanding the operator support. Under proper scaling of intermediate results, integer arithmetic alone suffices and expensive floating point can be dispensed with altogether. Zero-crossings are localized with a prairie-fire technique.

The above code, written for the CMU Grinnell environment (Vax/Unix/C, with IU testbed standards), has been distributed to other ARPA sites with comparable equipment.

The Grinnell Image Processor's ability to instantaneously pan over wide ranges of pixels has been exploited in another related way: to implement a fine-grained message-passing parallel shape-from-texture algorithm [10]. Although requiring some skill in design and debugging, it was surprising to be able to cast what was essentially a row-wise sort into a simple SIMD procedure. The algorithm, which generates vanishing lines based on texture density gradients, was executed on a real image with good result.

Acknowledgements

Graduate Fellow contributors to the Image Understanding Project include Terry Boulton, Paul Douglas, David Freudenstein, Hussein Ibrahim, David Lee, and Mark Moerdler. Our visionary programmer is Christian Fortuvel. Dan Sabbah (IBM, Yorktown) and Peter Selfridge (Bell Labs, Holmdel) contribute as departmental Research Computer Scientists.

References

1. Fortuvel, C. GRICON: Grinnell Convolution Package. Department of Computer Science, Columbia University, May, 1984.
2. Ibrahim, H.A.H. *Image Understanding Algorithms on Fine-Grained SIMD Machines*. Ph.D. Th., Department of Computer Science, Columbia University, Oct. 1984.
3. Ibrahim, H.A.H. Some Image Understanding Algorithms on Fine-Grained Tree-Structured SIMD Machines. Proceedings of the Workshop on Algorithm-Guided Parallel Architectures for Automatic Target Recognition, DARPA, July, 1984.
4. Ibrahim, H.A.H. The Connected Component Algorithm on the Non-Von Supercomputer. Proceedings of the Second Workshop on Computer Vision: Representation and Control, IEEE Computer Society, April, 1984.
5. Ibrahim, H.A.H., Kender, J.R., and Shaw, D.F. The Hough Transform Method on Fine-Grained Tree-Structured SIMD Machines. Proceedings of the ARPA Image Understanding Workshop, Sept., 1984. (These proceedings.)
6. Kender, J.R. Environmental Labelings in Low-Level Image Understanding. Proceedings of the 1983 International Joint Conference on Artificial Intelligence, Aug., 1983, pp. 1104-1107.
7. Kender, J.R. Surface Constraints from Linear Extents. Proceedings of the National Conference on Artificial Intelligence, Aug., 1983, pp. 187-190.
8. Kender, J.R., and Lee, D. The Information-Centered Approach to Optimal Algorithms Applied to the 2-1/2 D Sketch. Department of Computer Science, Columbia University, Sept., 1984. (Also these proceedings.)
9. Kender, J.R. *Artificial Intelligence Research Notes*. Volume : *Shape from Texture*. Pitman Publishing, Ltd., London, Accepted for publication, 1984.
10. Kender, J.R. Columbia's Vision Experience with Two Fine-Grained SIMD Architectures. Abstracts of Presentations at the Computer Architectures for Vision Workshop, DARPA, May, 1984, pp. 7-10.

THE SRI IMAGE UNDERSTANDING RESEARCH PROGRAM

M.A. Fischler
(Principal Investigator)

Artificial Intelligence Center, SRI International
Menlo Park, California 94025

Abstract

The SRI Image Understanding program is a broad effort spanning the entire range of machine vision research. Its three major concerns are: (1) to develop an understanding of the physics and mathematics of the vision process, (2) to develop a knowledge-based framework for integrating and reasoning about sensed (imaged) data, and (3) to develop a machine-based environment for effective experimentation, demonstration, and evaluation of our theoretical results, as well as providing a vehicle for technology transfer. This report describes recent progress in all three areas. In particular, we have shown that fractal functions are an effective tool for representing natural shapes and provide a good basis for recovering 3-D shape from the shading and texture in a single image. For scenes containing man-made objects, we have found ways of using straight edges to recover the 3-D orientation of surfaces from a single view, and reason about the shape of an object from partial information in multiple views. We have built a new and powerful LISP-machine-based environment for use in image understanding research, and are putting together high-performance systems for stereo compilation, feature extraction, and linear delineation.

1. INTRODUCTION

The goal of this research program is to obtain solutions to fundamental problems in computer vision, particularly to such problems as stereo compilation, feature extraction, linear delineation, and general scene modeling that are relevant to the development of an automated capability for interpreting aerial imagery and the production of cartographic products.

To achieve this goal, we are engaged in investigations of such basic issues as image matching, partitioning, representation, and physical modeling (shape from shading, texture, and optic flow; material identification; recovery of imaging and illumination parameters such as "vanishing points," "camera parameters," illumination source location; edge classification; etc.). However, it is obvious that high-level, high-performance vision requires the use of both intelligence and stored knowledge (to provide an integrative framework), as well as an understanding of the physics and mathematics of the imaging process (to provide the basic information needed for a reasoned interpretation of the sensed data). Thus, a significant portion of our work is devoted to developing new approaches to the problem of "knowledge-based vision." Finally, vision research cannot proceed without a means

for effective implementation, demonstration, and experimental verification of theoretical concepts; we have developed an environment in which some of the newest and most effective computing instruments can be employed for these purposes.

2. KNOWLEDGE BASED VISION: the Construction of an Expert System Control Structure for Stereo Compilation and Feature Extraction.

Our intent in this effort is to develop a system framework for allowing higher level knowledge to guide and integrate the detailed interpretation of imaged data by autonomous scene analysis techniques. Such an approach allows symbolic knowledge, provided by higher-level knowledge sources, to automatically control the selection of appropriate algorithms, adjust their parameters, and apply them in the relevant portions of the image. More significantly, we are attempting to provide an efficient means for supplying and using qualitative knowledge about the semantic and physical structure of a scene so that the machine-produced interpretation, constrained by this knowledge, will be consistent with what is generally true of the overall scene structure, rather than just a good fit to locally applied models.

An important component of our approach is to design a means for a human operator to simply and effectively provide the machine with a qualitative scene description, in the form of a semantically labeled 3-D "sketch." This capability for effective communication between a human and the machine about the three-dimensional world requires both appropriate graphics tools and an ability on the part of the machine for both spatial reasoning and some semantic "understanding." The importance of this work derives from the fact that a major difficulty in automating the image-interpretation process is the inability of current computer systems to deduce, from the visible image content, the general context of the scene (e.g., urban or rural; season of the year; what happened immediately before, and what will happen immediately after, the image was viewed by the sensor) - the knowledge base and reasoning required for such an ability is well beyond what the state of our art can hope to accomplish over (at least) the next 5 years. Thus, our work is intended to provide an interim means by which a human can supply, a task-oriented program, the high level overview it needs for its analysis, but cannot acquire by itself.

Two of our major integrative efforts are directly concerned with the knowledge-based vision problem:

One effort, integrating our work in stereo compilation and physical modeling, is the construction of a rule-based system with a library of processes and activities, which can be invoked to carry out specific goals in the domain of cartographic analysis and stereo reconstruction. This work is based on results described below, but the integrative framework is still being developed and will not be described in this report.

The second effort, described in a following section on feature extraction, is a restricted version of the concept discussed above (it employs contextual and semantic knowledge, but does not address the issues of qualitative reasoning nor 3-D spatial understanding).

3. DEVELOPMENT OF METHODS FOR MODELING AND USING PHYSICAL CONSTRAINTS IN IMAGE INTERPRETATION.

Our goal in this work is to develop methods that will first allow us to produce a sketch of the physical nature of a scene and the illumination and imaging conditions, and then permit us to use this physical sketch to guide and constrain the more detailed descriptive processes - such as precise stereo mapping.

Our approach is to develop:

- models of the relationship between physical objects in the scene and the intensity patterns they produce in an image (e.g., models that allow us to classify intensity edges in an image as either shadow, or occlusion, or surface intersection, or material boundaries in the scene),
- models of the geometric constraints induced by the projective imaging process (e.g., models that allow us to determine the location and orientation of the camera that acquired the image, location of the vanishing points induced by the interaction between scene and camera, location of a ground plane, etc.), and
- models of the illumination and intensity transformations caused by the atmosphere, light reflecting from scene surfaces, and the film and digitization processes that result in the computer representation of the image.

These models, when instantiated for a given scene, provide us with the desired "physical" sketch. We are assembling a "constraint-based stereo system" that can use this physical sketch to resolve the ambiguities that defeat conventional approaches to stereo modeling of scenes (e.g., urban scenes or scenes of cultural sites) for which the images are widely separated in either space or time, or for which there are large featureless areas, or a significant number of occlusions.

A summary of some of our current work in the area of modeling and using physical constraints is presented below.

Rectilinear Forms. Images of cultural scenes, such as building complexes, typically contain a significant amount of linear structure. We have developed an effective computational technique for recovering 3-D interpretations from a single 2-D image in many such cases. It works by finding a basis for a vector space suitable for quantifying spatial relations, while satisfying constraints imposed by linear features in the image. Given three image lines that are assumed to be perspective projections of

mutually orthogonal scene features, the method backprojects the lines into three-dimensional scene space, generating (potentially) all possible combinations of line orientations. It selects the combination that is "most orthogonal" by maximizing the triple product of three unit basis vectors, using the method of steepest descent. In general, two solutions are found, and the correct one can be chosen by relating the solutions to knowledge of the vertical direction. A more complete description of this work is presented in Barnard [1984a].

Inductive Approach. The technique discussed above has led us to investigate a new class of computational methods for the interpretation of single images. These methods constitute an inductive approach because they explicitly recognize that the available data (the image) are insufficient to make a well-founded logical interpretation; that is, many interpretations are possible, but only one is preferred. Specific prior models cannot account for the general power of such perception in the case of a human observer (although prior models are certainly used when available). To be truly general purpose, machine vision must be able to mimic this amazing human ability. The inductive approach selects interpretations that are "simplest" in some sense. While it does not preclude the use of specific prior models, it emphasizes the use of abstract generic models, such as parametric curves and surfaces. One measure of simplicity we have considered is based on information-theoretic considerations. This work will be described in a report by Barnard [1984b].

Optic Flow. In the optic flow paradigm, a moving observer is normally able to interpret a time sequence of images as an implicit description of a static scene. In principle, the images can be matched point-by-point and the motion of the observer can be deduced by exploiting the constraint that the scene is fixed. In practice, this is exceedingly difficult to achieve, both because point features are rare and because the methods are very sensitive to small matching errors.

We have developed an alternative optic-flow method that exploits the often available information about the rotation of the observer. Knowing the observer's rotation greatly simplifies the problem of matching successive images, but, since all the useful information that can be derived from the sequence is due to the translation of the observer, it does not significantly sacrifice generality. The major advantage to translation-only optic flow is that curvilinear image features can be matched by exploiting a constraint that is essentially the same as the epipolar constraint in stereo interpretation. This work is still in progress.

Spatial Reasoning from Line Drawings of Polyhedra. Construction of a three-dimensional "sketch" is one task faced by the user of an interactive image understanding expert system. An urban scene typically contains buildings and other objects that can be modeled as planar-faced polyhedra. An effective way for the user to create 3-D sketches from multiple views of such objects has been devised.

The system requires two or more line drawings of a polyhedral scene from arbitrary vantage points. These line drawings may be obtained from a freehand sketch, by tracing the edges in several photos, or from the output of an automatic edge detector. A "wireframe" model of the objects is obtained by

back-projecting the line drawings. Labels of solid or vacant space are then assigned to all spatial regions defined by the wireframe using an iterative constraint propagation algorithm. The result is a data structure that captures the volumetric structure of the objects depicted, which can then be used to support hidden-line elimination and other volumetric operations upon the object. This work is described in Strat [1984a].

Determining The Imaging Geometry from a Camera Transformation Matrix. Many scene analysis algorithms require knowledge of the geometry of the image formation process as a prerequisite to their application. When the imaging situation can be controlled or measured directly, the needed parameters can be determined; however, in the case of uncalibrated images, or photographs whose history is unknown, the necessary parameters are not available. In these cases, an alternate method of inferring the imaging situation from the correspondence between a small set of image and object points is required.

One approach has been to compute the imaging geometry directly from the constraints provided by the known data points. Partial information such as the camera's focal length or the location of the piercing point in the image can be used to reduce the number of data points needed. A second approach consists of two steps. First, the known data points are used to compute a 4x4 homogeneous coordinate transformation matrix that captures the entire transformation from object point to image point. An established technique for this computation involves the least squares solution of a set of simultaneous linear equations from six or more known correspondences. The goal of the second step is to derive the various parameters of the image formation process from the transformation matrix. This problem can be posed as a system of nonlinear equations whose solution had required iterative methods. Recently, Ganapathy [1984] published the first noniterative solution.

Research performed at SRI has also produced a noniterative solution (Strat [1984b]). By reasoning about the geometric constraints inherent in a camera transformation matrix, a simple, easily understood method of determining the various parameters is obtained. Through a series of geometric constructions, the camera's location and orientation, along with the piercing point and the relations between the focal length and scale factors, can be determined. The method relies purely on spatial reasoning about geometric constraints and does not involve an intuitively opaque matrix decomposition. Furthermore, its sensitivity to errors can be studied geometrically, allowing a clear understanding of the conditions that lead to inaccurate decompositions. The technique has been successfully applied to both synthetic imaging situations and real photographs.

4. STEREO COMPILATION: IMAGE MATCHING AND INTERPOLATION

We are implementing a state-of-the-art stereo system that produces dense range images given pairs of intensity images. We plan to use it both as a framework for our stereo research, and as the base component of an expert system concerned with 3-D

compilation.

There are five components of this stereo system: a rectifier, a sparse matcher, a dense matcher, an interpolator, and a projective display module. The rectifier accepts estimates of the parameters and distortions associated with the imaging process, the photographic process, and the digitization. These parameters are used to map digitized image coordinates onto an ideal image plane. The sparse matcher performs two-dimensional searches to find several matching points in the two images, which it uses to compute a relative camera model. The dense matcher tries to match as many points as possible in the two images. It uses the relative camera model to constrain the searches to one dimension, along epipolar lines. The interpolator computes a grid of range values by interpolating between the matches found by either the sparse or the dense matcher. The projective display module allows interactive examination of the computed 3-D model by generating 2-D projective views of the model from arbitrarily selected locations in space.

The current system, which runs on the VAX/11-780 in C, is described in Hannah [1984]. At present, the system produces relatively sparse 3-D information, even in its dense matching mode. Often 3-D data are required that are more closely spaced than can be provided by the stereo matching process. Further, there may be areas of the images that cannot be matched due to noise, insufficient information, and occlusions; this will produce holes in the dense matched data that must be filled in. In either case, interpolation is necessary to provide 3-D data between matched points.

Interpolation. We are currently exploring two different schemes for interpolation. One is a global approach, in which all of the 3-D information available is used to find the interpolated value for a given point. (This approach is described in Smith [1984].) The second approach is a local one, which only uses the data in the neighborhood of the point to be interpolated. The global approach produces a functional description that can be differentiated analytically to determine slope and other surface attributes; the local approach is most useful in the context of verifying the plausibility of the matches by comparing the data from the stereo images after projection onto this surface. The local approach is being used in the context of a hierarchical matching scheme described below.

Matching. In a parallel research effort employing our Lisp Machines, we are exploring a hierarchical technique for developing a regular, dense grid of matched points. This technique does appropriate warping of the images between each level of the hierarchy, to account for differences in perspective between the two images as predicted by the model. As a part of this effort, local interpolation techniques have been developed to fill in holes in the model before proceeding from level to level.

The Lisp Machine implementation includes a sophisticated terrain display package, which permits the user to interactively designate a flight path through the 3-dimensional model derived from a pair of images; the system then creates a "movie" (a sequence of either monocular or stereo views) of the terrain as the user "flies" along the path above the terrain. This package is useful not only for assessing the quality of the derived model,

but also for tasks in which a prediction of the appearance of the scene from arbitrarily specified points of view is desired, as when an observer is moving through mapped terrain. This work is described in Quam [1984].

Evaluation. We now have available, on our VAX (Testbed) and Lisp Machines, some of the most advanced stereo matching systems developed by the IU community. As a part of our stereo research effort, we plan to run several calibrated data sets through these systems to determine the relative strengths and weaknesses of the various methods, including area correlation, hierarchical warped matching, edge matching, and edge/intensity matching.

5. THE REPRESENTATION OF NATURAL SCENES

Our current research in this area addresses two related problems: (1) representing natural shapes such as mountains, vegetation, and clouds, and (2) computing such descriptions from image data. The first step towards solving these problems is to obtain a model of natural surface shapes.

A model of natural surfaces is extremely important because we face problems that seem impossible to address with standard descriptive computer vision techniques. How, for instance, should we describe the shape of leaves on a tree? Or grass? Or clouds? When we attempt to describe such common, natural shapes using standard representations, the result is an unrealistically complicated model of something that, viewed introspectively, seems very simple. Furthermore, how can we extract 3-D information from the image of a textured surface when we have no models that describe natural surfaces and how they evidence themselves in the image? The lack of such a 3-D model has restricted image texture descriptions to being ad hoc statistical measures of the image intensity surface.

Fractal functions, a novel class of naturally arising functions, are a good choice for modeling natural surfaces, because many basic physical processes (e.g., erosion and aggregation) produce a fractal surface shape and because fractals are widely used as a graphics tool for generating natural-looking shapes. Additionally, in a survey of natural imagery, we found that a fractal model of imaged 3-D surfaces furnishes an accurate description of both textured and shaded image regions, thus providing validation of this physics-derived model for both image texture and shading.

Progress relevant to computing 3-D information from imaged data has been achieved by use of the fractal model. A test has been derived to determine whether or not the fractal model is valid for a particular set of image data, an empirical method for computing surface roughness from image data has been developed, and substantial progress has been made in the areas of shape-from-texture and texture segmentation. Characterization of image texture by means of a fractal surface model has also shed considerable light on the physical basis for several of the texture-partitioning techniques currently in use and made it possible to describe image texture in a manner that is stable over transformations of scale and linear transforms of

intensity.

The computation of a 3-D fractal-based representation from actual image data has been demonstrated. This work has shown the potential of a fractal-based representation for efficiently computing good 3-D representations for a variety of natural shapes, including such seemingly difficult cases as mountains, vegetation, and clouds.

This research is expected to contribute to the development of (1) a computational theory of vision applicable to natural surface shapes, (2) compact representations of shape useful for natural surfaces, and (3) real-time regeneration and display of natural scenes. We also anticipate adding significantly to our understanding of the way humans perceive natural scenes.

Details of this work can be found in Pentland [1983 and 1984].

6. FEATURE EXTRACTION: SCENE PARTITIONING, AND LABELING

Our efforts in image partitioning and labeling have advanced along two fronts: we have developed a goal-directed texture-based segmentation algorithm and have studied knowledge-based control concepts required to integrate this with other image feature-extraction techniques.

The SLICE goal-directed segmentation system combines knowledge of target textures or signatures with knowledge of background textures by using histogram-similarity transforms. Regions of high similarity to a target texture and of low similarity to any negative texture examples are found. This use of semantic knowledge during the segmentation process improves segmenter performance and focuses segmentation activity on material types of greatest interest. (The system can also be used for goal-independent texture segmentation by omitting the similarity-transform computations.) Development of this segmentation technique is essentially complete; all that remains is to integrate it into the more general feature-extraction system described below. Performance of the SLICE segmentation algorithm is documented in Laws [1984].

The KNIFE (knowledge-based interactive feature-extraction) system is intended to solve problems in image segmentation, feature extraction, material identification, and feature classification. (Image segmentation and feature extraction partition an image into meaningful units; material identification and feature classification label those units.) Experience has shown that these tasks cannot be carried out adequately in isolation. Image segmentation, for instance, cannot produce a meaningful partitioning unless it is guided by semantic criteria from material identification and feature classification.

The KNIFE feature-extraction system will combine a data base of recognition rules (using shape, texture, and context) with recursive segmentation and other techniques to find and label scene features. Initially selected image regions, based on image brightness and texture, are resegmented and refined to locate recognizable objects (e.g., roads, fields, and buildings). The control process assigns initial labels for each region, and then recursively analyzes those regions that might contain useful substructure. The choice of regions to split or merge is influenced

by analysis goals rather than solely by statistical properties of the image data. The segmentation and interpretation will thus proceed at unequal rates or to different depths in separate scene regions, with differing types of knowledge applied at successive stages in the analysis. Objects detected by other means (user interaction or direct object recognition) may override the normal interpretation cycle.

We are concentrating our development efforts on goal-directed recursive segmentation and on related display, query, and editing tools. Among these tools are display of input images and segmentation maps; readout of region descriptions and relationships; and commands for interactively designating, splitting, merging, and classifying regions.

The control process is a production system that looks for applicable rules in the rule base. Such rules will be placed on a prioritized queue of tasks to be performed. When executed, they may query the user, invoke image analysis subsystems, or affect the behavior of the control process itself.

Besides the rule base and the input or derived imagery, the system will have two principal data structures. These are the sketch data base, and the prototype data base.

The sketch data base serves as the system blackboard, storing all the information relevant to the current image. The prototype data base will be a semantic network with nodes storing object properties and pointers to image examples.

The system is being developed on the VAX-based SRI Image Understanding (IU) Testbed. The basis for the system's data analysis capabilities will be the body of software currently accumulated in the testbed and other programs now being developed, such as the SLICE goal-directed segmentation system discussed above.

7. LINEAR DELINEATION AND PARTITIONING

A basic problem in machine-vision research is how to produce a line sketch that adequately captures the semantic information present in an image. (For example, maps are stylized line sketches that depict restricted types of scene information.) Before we can hope to attack the problem of semantic interpretation, we must solve some open problems concerned with direct perception of line-like structure in an image, and with decomposing complex networks of line-like structures into their primitive (coherent) components. Both of these problems have important practical as well as theoretical implications.

For example, the roads, rivers, and rail-lines in aerial images have a line-like appearance. Methods for detecting such structures must be general enough to deal with the wide variety of shapes they can assume in an image as they traverse natural terrain.

Most approaches to object recognition depend on using the information encoded in the geometric shape of the contours of the objects. When objects occlude or touch one another, decomposition of the merged contours is a critical step in interpretation.

We have made significant progress in both the delineation and the partitioning problems. Our work in delineation (Fischler

and Wolf [1983]) is based on the discovery of a new perceptual primitive that is highly effective in locating line-like (as opposed to edge-like) structure.

One approach to decomposing linear structures into coherent components (Fischler and Bolles [1983]) is based on the concept that perception is an explanatory process - acceptable precepts must be associated with explanations that are believable: They must be complete (i.e., they explain all the data), simple (i.e., both concise and of limited complexity), and stable (i.e., they must not change under small perturbations of either the imaging conditions or the decision algorithm parameters).

A second approach to the partitioning problem, which also addresses the problem of qualitative matching of linear structures (Smith and Wolf [1984]), focuses on the concept of simplicity as the basis for making perceptual decisions. Given a set of primitives as the basis for description, each possible description of a set of data is evaluated as to how accurately it describes the data and how "long" a description is required (a natural conversion from accuracy to descriptive length is provided). The shortest description is chosen as being correct.

These new delineation and partitioning algorithms have produced excellent results in experimental tests on real data. Our continuing work in this area focuses on theoretical, as well as performance, issues.

8. COMPUTING ENVIRONMENT FOR IU RESEARCH

Previous reports (e.g., Hanson and Fischler [1981]) describe the VAX 11/780 testbed environment we created for evaluation, demonstration, and transfer of IU technology. A significant recent addition to this system is based on the Symbolics 3600 LISP machine. Documentation of this new system is still incomplete, but as noted in section four of this report (Stereo Compilation: matching), applications recently considered beyond the state-of-the-art on comparably priced hardware, have already been programmed and demonstrated (Quam [1984]).

Acknowledgement

The following researchers have contributed to the work described in this report: H. Baker, S. Barnard, R.C. Bolles, M.A. Fischler, M.J. Hannah, A.J. Hanson, D.L. Kashtan, K. Laws, O. Firschein, A.P. Pentland, L.H. Quam, G.B. Smith, T. Strat, and H.C. Wolf.

References

- S.T. Barnard, *Choosing a Basis for Perceptual Space*, *Proc. IEEE Workshop on Computer Vision: Representation and Control*, Annapolis, Maryland, (April 1984) (to appear in *Computer Vision, Graphics, and Image Processing*).
- S.T. Barnard, "An Inductive Approach to Figural Perception," Technical Note (in preparation), Artificial Intelligence Center, SRI International, Menlo Park, California (1984).

M.A. Fischler and H.C. Wolf, "Machine Perception of Linear Structure," *Proc. 8th Int. Jnt. Conf. on Artificial Intelligence*, Karlsruhe, West Germany, pp. 1010-1013 (8-12 August 1983).

M.A. Fischler and R.C. Bolles, "Perceptual Organization and the Curve Partitioning Problem," *Proc. 8th Int. Jnt. Conf. on Artificial Intelligence*, Karlsruhe, West Germany, pp. 1014-1018 (8-12 August 1983).

S. Ganapathy, "Decomposition of Transformation Matrices for Robot Vision," *IEEE*, pp. 130-139 (1984).

M.J. Hannah, "Description of SRI's Baseline Stereo System," Technical Note (in preparation), Artificial Intelligence Center, SRI International, Menlo Park, California (1984).

A.J. Hanson and M.A. Fischler, "The DARPA/DMA Image Understanding Testbed" *Proc. DARPA Image Understanding Workshop*, Palo Alto, California, (15-16 September 1982).

K.I. Laws, *Goal-Directed Textured-Image Segmentation*, Technical Note 334, Artificial Intelligence Center, SRI International, Menlo Park, California (September 1984).

A.P. Pentland, "Fractal-Based Description," *Proc. 8th Int. Jnt. Conf. on Artificial Intelligence*, Karlsruhe, West Germany, pp. 973-981 (8-12 August 1983).

A.P. Pentland, "Shading Into Texture," *Proc. Nat. Conf. on Artificial Intelligence*, Austin, Texas, pp. 269-273 (6-10 August 1984). Also in these proceedings.

L.H. Quam, "Hierarchical Warp Stereo," *Proc. DARPA Image Understanding Workshop*, New Orleans, Louisiana, (3-4 October 1984). These proceedings.

G.B. Smith, *A Fast Surface Interpolation Technique*, Technical Note 333, Artificial Intelligence Center, SRI International, Menlo Park, California (August 1984). Also in these proceedings.

G.B. Smith and H.C. Wolf, *Image-to-Image Correspondence: Linear-Structure Matching*, Technical Note 331, Artificial Intelligence Center, SRI International, Menlo Park, California (July 1984). Also in these proceedings.

T.M. Strat, "Spatial Reasoning from Line Drawings of Polyhedra," *Proc. IEEE Workshop on Computer Vision: Representation and Control*, Annapolis, Maryland (April 1984). Also in these proceedings.

T.M. Strat, "Recovering the Camera Parameters from a Transformation Matrix," *Proc. DARPA Image Understanding Workshop*, New Orleans, Louisiana (October 1984). These proceedings.

MIT PROGRESS IN UNDERSTANDING IMAGES

T. Poggio and the staff

The Artificial Intelligence Laboratory, Massachusetts Institute of Technology

The gap between the image arrays and the high level descriptions that are required for many visual tasks is too wide to be bridged in a single step. Intermediate steps are necessary, leading to several representations of the visible world. Our work to date has focused primarily on the initial representations of low-level vision up to the 2.5-D sketch that encodes information about the 3-D surfaces and their properties. We are now turning our efforts to the integration of different sources of information and to various aspects of the general problem of deriving a powerful symbolic representation of the world from image intensities. In this report we review our recent work on early vision, starting from the perspective of regularization analysis which is providing a new and powerful theoretical framework for most of early vision. In particular, we will discuss our progresses in edge detection, multiple scale methods, computation of motion, stereo algorithms, multigrid algorithms and surface reconstruction across discontinuities. We also describe some of our work on higher level vision, including shape representation, object recognition and the analysis of spatial relations among objects and object parts.

1. Regularization: the New Approach

Until recently, researchers in vision had little common theoretical framework to call upon. It is true that significant progress has been made in recent years toward solving some problems of early vision and implementing those solutions in working algorithms. But the methods, the tools, and the techniques were specific to each problem and had to be invented fresh on each occasion.

A recent theoretical development promises to improve on this situation. We now believe that most of the early vision problems can be "solved" using *regularization analysis*. This new approach leads to a specific, powerful class of algorithms for most problems in vision, and to parallel, fine-grained, local interconnections hardware for implementing these algorithms efficiently (Poggio and Torre, 1984; Poggio and Koch, 1984). As we will see, regularization analysis is far from being a one-shot solution to early vision. A physical analysis of any specific problem and of its generic constraints are critically important. Examples of domains of analysis that are required and can be exploited in regularization analysis are the physics of image formation and multiresolution analysis of images. In addition any specific module of early vision requires an analysis of its specific physical constraints.

1.1. Ill-posed Problems and Regularization Analysis

In 1923, Hadamard defined a mathematical problem to be *ill-posed* when its solution

- Does not exist.
- Is not unique.
- Does not depend continuously on the initial data, or said another way, the solution is not robust in the face of noise.

Most early vision problems are ill-posed in Hadamard's sense. There are three reasons for this. First, most early vision problems have no unique solution. Second, their solutions do not depend continuously on the data. And third, most early vision problems are inverse problems, and we know that most inverse problems are ill-posed. We have formally shown that several low-level vision problems such as edge detection, motion measurement, stereo matching and surface interpolation, are ill-posed in Hadamard's sense (Poggio and Torre, 1984).

Our optimism about the prospect of solving much of early vision flows from recent advances that American and Russian mathematicians have made in developing rigorous regularization methods for "solving" ill-posed problems. The basic idea behind these regularization techniques is to restrict the space of acceptable solutions by choosing a function that minimizes an appropriate functional. The mathematics involved in regularizing ill-posed problems leads to choices that depend fundamentally on a physical analysis of the generic constraints on the problem. It has long been recognized that the identification of appropriate physical constraints is a necessary prerequisite to the formulation of early vision problems in a way that is well-defined and solvable. In fact, some vision problems such as shape from shading, surface interpolation and motion measurement have previously been formulated precisely in the form required by standard regularization methods. This common theoretical framework allows us to apply these rigorous methods to many other ill-posed problems in vision.

1.2. Standard Regularization Techniques for Early Vision

Early vision can best be defined as inverse optics. Its main goal can be viewed as the solution to inverse problems. For example, in many problems, one seeks a solution z , given data y , such that $Az = y$. To apply regularization methods, one must first choose a set of norms $\|\cdot\|$ (usually quadratic) and a *stabilizing functional* $\|Pz\|$. The problem is then restated as the following variational problem: find a solution z such that functional (1) is minimized,

$$\|Az - y\|^2 + \lambda \|Pz\|^2, \quad (1)$$

The first term captures the closeness of the solution to the data. The second term captures the degree of regularization of the solution, and generally embodies the additional physical constraints necessary to solve the problem. The regularization parameter λ controls the compromise between these two factors. The regularization techniques that we are presently extending to early vision provide ways to determine the best λ . They also provide results about the form of the *stabilizing functional* P that ensure uniqueness of the result and rapid convergence of the computation.

We have recently shown that our work on the problems of shape from shading, computation of motion, and surface reconstruction can be reformulated as instances of the main regularization method. We will discuss some of these problems in more detail in the next sections. We have also applied regularization methods to the problem of edge detection, obtaining an explicit form for the optimal filter (Poggio et al., 1984). We will briefly review this and other work in the next section. Other problems such as stereo depth determination, and structure from motion can also be approached in terms of regularization analysis. At present we are working on several of these problems.

In summary, the concept of ill-posed problems and the associated regularization theories provide a powerful theoretical framework for solving many of the problems of early vision. This new perspective justifies the use of specific variational principles for solving certain problems and suggests how to approach many other early vision problems. Most importantly, it provides a link between the ill-posed nature of early vision problems and the computational structure of the algorithms for solving them. We are exploiting this link by designing fine grained hardware to efficiently implement these algorithms.

So far we have found two powerful classes of algorithms for solving variational problems of the type indicated by equation (1). They consist of filtering operations and of multigrid methods. Multigrid methods are a general and efficient method for solving quadratic variational problems of the type of equation (1). When the data are dense and given on a regular grid, a simpler method can be used, for appropriate boundary conditions: the solution can be computed by convolving the data y with a suitable filter (see Poggio and Torre, 1984). Thus, the structure of these algorithms leads directly to parallel, fine-grained hardware with local interconnections of the sort used in the Connection Machine. We are presently beginning to explore how to implement regularization methods efficiently in the Connection Machine architecture.

2. Edge Detection

We have recently applied regularization techniques to another classical problem of early vision - edge detection. Edge detection, intended as the process that attempts to detect and localize significant changes of intensity in the image can be regarded as a problem of numerical differentiation (Torre and Poggio, 1984). Notice that differentiation is a common operation in early vision and is not restricted to edge detection. The problem is ill-posed because the solution does not depend continuously on the data.

The intuitive reason for the ill-posed nature of the problem can be seen by considering a function $f(x)$ perturbed by a very small "noise" term $\epsilon \sin \Omega x$. $f(x)$ and $f(x) + \epsilon \sin \Omega x$ can be arbitrarily close for very small ϵ , but their derivatives may be very different if Ω is large enough. This simply means that a derivative operation "amplifies" high-frequency noise.

In 1-D, numerical differentiation can be regularized in the following way. Let the image model be $y_i = f(x_i) + \epsilon_i$, where y_i is the data and ϵ_i represent errors in the measurements. We want to estimate f' . We chose a regularizing functional $\|Pf\| = \int (f''(x))^2 dx$, where f'' is the second derivative of f . One regularization method would be to find a function f that minimizes the functional $\|Pf\|$. This method assumes no noise in the data, and is equivalent then to using interpolating cubic splines for differentiation. Another regularizing method, which is more natural since it takes into account errors in the measurements, leads to the variational problem of minimizing (see Rheinsch, 1967)

$$\sum (y_i - f(x_i))^2 + \lambda \int (f''(x))^2 dx. \quad (2)$$

Poggio, Voorhees and Yuille (1984) have shown (a) that the solution of this problem f can be obtained by convolving the data y_i (assumed on a regular grid and satisfying appropriate boundary conditions) with a convolution filter R , and (b) that the filter R is a cubic spline with a shape very close to a Gaussian and a size controlled by the regularization parameter λ . Differentiation can then be accomplished by convolution of the data with the appropriate derivative of this filter. The optimal value of λ can be determined for instance by cross validation and other techniques. This corresponds to finding the optimal scale of the filter (see Poggio and Torre, 1984).

These results can be directly extended to two dimensions to cover both edge detection and surface interpolation and approximation. The resulting filters are very similar to two of the gaussian-based edge detection filters derived and extensively used in recent years (Marr and Hildreth, 1980; Canny, 1983; see Torre and Poggio, 1984). The present derivation is, however, more general and rigorous: the filter follows naturally from regularizing the ill-posed problem of numerical differentiation for regularly spaced image data.

In the area of edge detection, the problem of which differential operators should be used after the filtering operation has been analyzed theoretically and with computer experiments by Torre & Poggio and co-workers (1984). In particular, they have derived relationships among several 2-D differential operators and characterized the relation between the Laplacian and the second directional derivative along the gradient. In addition they have studied the properties of the critical points of the differential operators and characterized the geometrical and topological properties of the zero-crossings (and level-crossings) of differential operators in terms of transversality and Morse theory.

3. Multiple Scale Analysis

As we have seen, differential operations on sampled images require the image to be first smoothed by filtering. The filtering operation introduces an arbitrary parameter - the scale of the filter, e.g., the standard deviation for the Gaussian filter, which is strictly connected, as we saw in the previous section, with the regularization parameter λ . In computer vision, the necessity of considering several scales of filtering was realized quite early on. This was supported by evidence suggesting the presence of filters of several sizes in the human visual system (Rosenfeld, 1982). More recently, Witkin (1983; see also Stansfield, 1980) introduced a scale-space description of zero-crossings which gives the position of the zero-crossing across a continuum of scales, i.e., sizes of the Gaussian filter (parametrized by the σ of the Gaussian). The signal—or the result of applying a linear (differential) operator to the signal—is convolved with a Gaussian filter over a continuum of sizes of the filter. Zero- or level- crossings of the filtered signal are contours on the $x - \sigma$ plane and surfaces in the x, y, σ space. Witkin proposed that this concise map can be effectively used to obtain a rich and qualitative description of the signal. Yuille and Poggio (1983a, 1983b) have established interesting relationships between multiresolution analysis, the Gaussian filter and zeros of differential operators. Their main results are two theorems:

(a) Zero- and level-crossings of an image filtered through a linear differential operator of the Gaussian filter have nice scaling properties, i.e., a simple behavior of zero-crossings across scales, with several attractive properties for further processing. Zero-crossings are not created as the scale increases. The nice scaling behavior is a characteristic property of the Gaussian filter and only of the Gaussian filter (see also Babaud, Witkin and Duda, 1983).

(b) The map of the zero-crossings across scales determines the signal uniquely for almost all signals in the absence of noise. The scale maps obtained by Gaussian filters is thus a complete representation of the image. This result applies to level-crossings of any arbitrary linear differential operator of the Gaussian, since it applies to functions that

obey the diffusion equation (the zero-crossings are then a unique characterization modulus the null space of the differential operators and provided there are at least two zero-crossings contours).

The first result sheds some light on the properties of zero-crossings and level-crossings at different scales, assuming the Gaussian filter. It also supports the use of the Gaussian filter in a multiresolution edge detection scheme. The second result implies that scale-space fingerprints are complete primitives, that capture the whole information in the signal and characterize it uniquely. Reconstruction of the signal is, of course, not the goal of early signal processing. Symbolic primitives must be extracted from the signals and used for later processing. Subsequent processes can therefore work on this more compact representation instead of the original signal (see Asada and Brady, 1984).

The second theorem has theoretical interest in that it answers the question of what information is conveyed by the edges identified with zero- and level-crossings of multiscale Gaussian filtered signals. It is furthermore interesting that this complete representation happens to coincide with the basic scheme for edge detection discussed earlier. From this point of view it can be argued that the fingerprint representation makes explicit exactly the information that is needed on physical grounds, i.e., it makes explicit edges in the image. We are now attempting (H. Voorhees, T. Poggio and A. Yuille) to attack again the problem posed by the primal sketch — of labeling changes in intensity in terms of the underlying surface properties — exploiting the use of fingerprints. The idea is to identify a small number of primitive image intensity features — such as step edges and roof edges — and partially label them in terms of the properties of the underlying physical surfaces, distinguishing for instance shadows from occlusion boundaries. The initial success of a similar attempt in the realm of 2-D shape representation by Asada and Brady (1984), which we will describe later, is encouraging.

It may be asked at this point what the correct sequence is for the two steps of differentiation and filtering. For linear operators the order is of course immaterial, since they commute. This is not the case for nonlinear operators, such as the directional derivative along the gradient. The regularization argument for the filtering step implies that filtering at one scale must precede the differentiation operation. The computation of different scales requires filtering at a range of resolutions after differentiation. The reason is that the theorems of Yuille and Poggio (1983a, 1983b) hold true even for the identity operator, but are not necessarily valid if filtering is performed before a nonlinear differential operation. In particular, Gaussian scaling after the nonlinear directional derivative along the gradient does not have a nice scaling behavior. Thus filtering as a regularizing operator must be performed first at one scale and filtering at different scales must be performed after the differential operation. For linear differential operators, this is equivalent to a multiscale filtering either before, after, or together with the differential operation (e.g. the Laplacian of the Gaussian).

4. Computation of Motion

In the area of visual motion analysis, we have developed a zero-crossing based method for computing the projected two-dimensional velocity field from a changing image. The method allows arbitrary three-dimensional surfaces undergoing general motion in space. The measurement of motion is, in general, an ill-posed problem, because the solution is not unique: there are infinitely many two-dimensional velocity fields consistent with a given dynamic image. This problem can be approached using the standard techniques of regularization analysis mentioned earlier (see also Poggio and Torre, 1984). Hildreth (1984) has developed a velocity field algorithm consisting of two main steps: (1) initial motion measurements are made at the locations of zero crossings, and provide the component of velocity in the direction perpendicular to the local orientation of the zero-crossing contours

(we denote this component by the function $v^\perp(s)$, where s is a curve parameter); and (2) a velocity field is computed that is consistent with $v^\perp(s)$ and which exhibits the least amount of variation along the zero-crossing contours. In particular, the velocity field $V(s)$ is computed that minimizes the measure of variation given by the integral $\int \left| \frac{\partial V}{\partial s} \right|^2 ds$ along the contour (this is an instance of the second regularization method described in Poggio and Torre, 1984). Except in the case of an infinite straight line, there exists a unique velocity field that minimizes this measure.

This velocity field algorithm has been implemented and tested on both synthetic and natural images. The synthetic images consisted of two types: (1) ideal contours undergoing a known motion, in which the measurements of $v^\perp(s)$ were computed analytically, and (2) natural images undergoing an artificial movement. It can be shown analytically (Yuille, 1983) that the computed velocity field of least variation will be equivalent to the true projected velocity field when the following relationship is satisfied along the contour:

$$T \cdot \frac{\partial^2 V}{\partial s^2} = 0$$

where T is the local tangent vector along the contour. If we assume orthographic projection of the scene onto the image plane, there are at least two general classes of motion for which the above relationship is satisfied: (1) pure translation of arbitrary objects through space, and (2) rigid rotation and translation of three-dimensional objects whose edges are straight lines. Empirical analysis of these classes of motion verifies the correctness of the velocity field derived from this algorithm (Hildreth 1984).

For the case of natural motion sequences, it was found that there can be considerable error in the measurement of the perpendicular velocity components, $v^\perp(s)$. This led to a reformulation of the algorithm in such a way that the computed velocity field only approximately satisfies $v^\perp(s)$. In particular, the algorithm minimizes the following expression:

$$\int \left| \frac{\partial V}{\partial s} \right|^2 ds + \lambda \int [V \cdot u^\perp(s) - v^\perp(s)] ds$$

where $u^\perp(s)$ is the unit vector in the direction perpendicular to the contour. The first term describes the variation in the velocity field, and the second describes how well the computed velocity field satisfies the image measurements given by $v^\perp(s)$. This formulation of the motion measurement computation is precisely of the type required by standard regularization methods, as shown earlier in equation (1). Empirical studies have shown that velocity field algorithms derived from this formulation are far more robust in the presence of error in the initial motion measurements derived from the changing image.

For the case of aerial photographs, the entire scene can be treated essentially as a single surface, undergoing a single motion. The relative movement of objects on the ground is very small compared to their overall displacement with respect to the airplane. In general, however, a scene will contain multiple objects undergoing different motions, with sharp discontinuities in the velocity field along object boundaries. The detection of these discontinuities is especially important for algorithms such as the one described here, which compute a smoothly varying pattern of movement. We have begun to explore possible algorithms to detect motion discontinuities which search for sudden changes in the sign or magnitude of the initial perpendicular components of velocity. These algorithms provide an indication of the location of the discontinuities prior to the combination of the measurements of $v^\perp(s)$ to compute the full two-dimensional velocity field. The incorporation of a discontinuity detection algorithm with the subsequent velocity field algorithm will lead to a more robust measurement of motion for scenes with multiple objects undergoing different movements.

5. Stereo Algorithms

We are presently developing a regularization solution to stereo matching, about which we will report soon (Yuille and Poggio). Here we will describe two other very recent approaches to the problem of stereo matching. The first algorithm, which is a considerable evolution of the original stereo theory of Marr and Poggio (1979), has been developed by Nishihara (1984) with the goal of achieving a high speed, noise tolerant stereo matcher. This matching scheme, based on patchwise correlation (between filtered images), can be shown to be a special case of a more general variational principle that can be derived with standard regularization methods. The second approach by Yuille and Poggio (1984) leads to a generalization of the *ordering constraint* (Baker, 1982) that captures several powerful constraints for solving the correspondence problem of stereo.

5.1. A Fast, Noise-Tolerant Stereo System

Nishihara has developed an approach to solving the binocular-stereo-matching problem which places special emphasis on the practical issues of noise tolerance, reliability, and speed. It is strongly influenced by Marr and Poggio's zero-crossing theory, but differs from recent implementations in the way zero-crossing information is used to drive the matching and in the product the matcher is designed to produce.

Four design objectives have guided Nishihara's study. The first is *noise tolerance*. We want to understand how matching can be accomplished in the presence of moderate to large noise levels which occur anytime surface contrast is low compared with sensor and inter-image distortions. The second objective is to achieve *competent performance* for at least one of the three kinds of stereo measurements—volume occupancy, range, and location of elevation discontinuities (Nishihara and Poggio, 1984). The third objective has been to operate at a *practical speed* using existing computer technologies. The emphasis in this work has been to streamline the computation to increase speed and use processing resources efficiently. This has forced a careful consideration of the relative cost of producing a measurement in different ways vis-a-vis its contribution to the final product of the algorithm.

With these design considerations as a base, Nishihara has designed a binocular-stereo-matching algorithm for making rapid visual range measurements in noisy images. This technique is developed for application to problems in robotics where noise tolerance, reliability, and speed are predominant issues. A high speed pipelined convolver for preprocessing images and an *unstructured light* technique for improving signal quality are introduced to help enhance performance to meet the demands of this task domain. These optimizations, however, are not sufficient. A closer examination of the problems encountered suggests that broader interpretations of both the objective of binocular stereo and of the zero-crossing theory of Marr and Poggio are required. This research has been restricted to the problem of making a single primitive surface measurement. For example, to determine whether or not a specified volume of space is occupied, to measure the range to a surface at an indicated image location, or to determine the elevation gradient at that position. In this framework a subtle but important shift is made from the explicit use of zero-crossing contours (in band-pass filtered images) as the elements matched between left and right images, to the use of the signs of the convolution between zero-crossings. With this change, a simpler algorithm is obtained with a reduced sensitivity to noise and a more predictable behavior. The PRISM system incorporates this algorithm with the unstructured light technique and a high speed digital convolver. It has been used successfully by both R. Brooks and K. Ikeuchi as a sensor in path planning and bin picking systems respectively.

The PRISM system incorporates an efficient module for determining the two-dimensional displacement, τ_0 , between patches out of the left and right images. This *near/far* module does not do point by

point search. Instead, a single correlation measurement is made at a test disparity (provided as input) and a determination is made as to whether the correlation peak can be near by (within half the excitatory diameter of the JOG operator). If there is a positive result, several additional correlation measurements are made at neighboring disparities to determine the shape of the correlation function over the test disparity. From this an estimate is made for the disparity at which the correlation peak occurs. The name of the module comes from the work of G. Poggio and Fisher who described a class of neurons in primate visual cortex sensitive to either near or far disparities.

The principal surface parameter is distance from the cameras which manifests itself as a translational disparity between corresponding patches from the two images. We are also able to correlate against parameters other than translational disparity. For example, an elevation gradient on the physical surface viewed can be measured by correlations against the compressive and shear distortions. These distortions are introduced between the left and right images by horizontal and vertical elevation gradients.

5.2. A Generalized Ordering Constraint for Stereo

The problem of stereo matching is ill-posed and underdetermined; constraints are needed to make the solution unique, and to reduce the search problem among possible matches.

Marr and Poggio (1979) originally identified two constraints: (1) *uniqueness*, that is, an element in one image in general only corresponds with a single element in the other image, and (2) *continuity*, that is, stereo disparity varies smoothly almost everywhere in the image. These constraints are powerful because they do not depend on the specific properties of the scene but on general properties of the stereo geometry. Marr and Poggio (1979) proposed a stereo matching algorithm, further developed by Grimson (1981, 1984), which incorporates the uniqueness and continuity constraints to match zero-crossing descriptions computed at different scales.

An *ordering constraint along epipolar lines* has been exploited, both implicitly and explicitly, in several computer algorithms for stereo matching (for example, Baker and Binford, 1981) as a special instance of the continuity constraint. Epipolar lines in the two images are lines on which corresponding points lie. The projections of a point P in space lie on the plane defined by P and the two camera foci and, as a consequence, on the two lines defined by the intersection of this plane with the two image planes. This implies that the matching problem can be reduced to a one-dimensional search if the epipolars are known. Most algorithms assume that the epipolar geometry is known (from a known camera geometry) and that the images are registered. Furthermore, the *ordering* of edges or other features is usually preserved by stereo projection along epipolar lines (that is, if feature A is to the left of feature B in the left stereo image, then this spatial relationship is maintained in the right stereo image). The ordering constraint along epipolar lines follows from the continuity of surfaces and the assumption of opacity. As originally suggested by Baker (1982) the ordering constraint is violated in some situations (in the "forbidden zone"). The forbidden zone associated with each point of the visible surface is a set of points in space that would have images violating the ordering constraint. If any point in the forbidden zone would be connected to the first point by an opaque surface the two images would "see" opposite sides of the surface.

This *ordering constraint* can be exploited to reduce the complexity of the search for matching features, and to eliminate false matches. There are, however, situations in which the images are not precisely registered. Furthermore, physical edges are inherently two-dimensional (a property that is not exploited by the epipolar ordering constraint). It is therefore natural to ask whether the ordering constraint can be generalized away from epipolar lines. Yuille and Poggio (1984) have shown that it is indeed possible to generalize the ordering constraint.

The analysis of Yuille and Poggio considers a simple stereo geometry: it begins by proving a simple relationship between the two images of a 3D curve that leads to a generalization of the classical ordering constraint. This relationship allows one to identify special points in the images that correspond uniquely to the same physical point in the object curve. The Generalized Ordering Constraint (GOC) implies several of the specific constraints listed by Baker et al. (1983) (see the crossproduct constraint), Mayhew and Frisby (1981) (see their *figural continuity constraint*), and Ohta and Kanade (1983). The ordering constraint breaks down when the object curve enters the *forbidden zone*. From this analysis, Yuille and Poggio outline an algorithm based on matching contours. From a single contour the algorithm retrieves the viewing parameters and unambiguously matches points along the contour using the generalized ordering constraint. Tiffany is now developing an algorithm based on this constraint.

6. Multigrid Algorithms for Regularization Analysis

6.1. Background and test report of surface reconstruction algorithm

We have investigated the computation of visible-surface representations, and related problems. As was described in previous reports, we have considered the efficient reconstruction of visual surfaces from scattered depth constraints using multiresolution iterative processing. The theoretical basis of the schemes has recently been tested on several sets of data — on data which was generated synthetically, on natural image data preprocessed by a photometric stereo method and two stereopsis algorithms, as well as data from Brou's laser rangefinder system (Terzopoulos, 1984a). The results of this extensive testing, coupled with the fact that the surface reconstruction code has been employed in other research projects in the laboratory, attests to the computational efficiency and robustness of the surface reconstruction algorithm.

6.2. New multiresolution algorithms

As was argued by several authors (Grimson, 1981; Brady and Horn, 1982 and especially Terzopoulos, 1984c), many early vision problems can be posed as variational principles. This is substantiated further by Poggio and Torre's (1984) view of problems in early vision as mathematically ill-posed problems which can be transformed into well-posed variational principles by regularization techniques. An attractive feature of these types of variational principles, once discretized, is that their solutions can be computed by local, iterative algorithms, which can be executed by many processors arranged in locally-connected networks or grids. Given only local processing capabilities, however, the essential global properties (e.g. (piecewise) smoothness, consistency, minimal energy, etc.) of the desired solutions must be satisfied indirectly by propagating visual information across grids, through iteration. Substantial computational inefficiency can result since grids can become extremely large in machine vision applications. Multiresolution iterative processing can overcome this inefficiency, as demonstrated by the application of multigrid methods (Hackbusch and Trottenberg, 1982) to visual surface reconstruction.

Exploring further the idea of applying multigrid methods in early vision, Terzopoulos (1984d) has developed and implemented efficient multiresolution iterative algorithms for other early vision problems; in particular, the computation of lightness, shape from shading, and optical flow from images. These new algorithms are based on the theoretical work of Horn and his colleagues; however, there are certain interesting novelties aside from the fact that the algorithms compute consistent visual representations at multiple scales. Notably, the lightness algorithm is locally parallel and iterative whereas Horn's involved convolution with a (global-support) Green's function, the shape from shading algorithm works in conjunction with the multiresolution surface reconstruction algorithm to provide a

representation of the surface in depth, and the optical flow algorithm can handle flow discontinuities at known occluding boundaries. The multiresolution algorithms have been tested on synthetic images, and all three are significantly more efficient than the single level versions (between one and two orders of magnitude).

This approach provides a general, efficient and powerful method for solving all vision problems that can be formulated in terms of quadratic regularization principles. We are now considering the eventual implementation of these multiresolution algorithms on our Connection Machine.

7. Regularization Theory of Discontinuities

As we discussed in section 1, many early vision problems can be characterized as ill-posed problems and treated by regularization methods such as that represented by eq. (1). A standard class of stabilizing functionals are so called Tikhonov stabilizers that typically impose smoothness conditions on the possible solutions (e.g., restricts them to be members of Sobolev spaces). This is sufficient to adequately solve a number of ill-posed problems in physics, and it applies in an approximate sense in vision inasmuch as the coherence of matter tends to produce smooth surfaces relative to the viewing distance at certain scales (of course, regularization theory is not restricted to Tikhonov stabilizers!).

A notable complication which arises repeatedly in early vision problems, however, is the necessity of dealing with discontinuities. Discontinuities are ubiquitous at all stages of visual processing taking part in the representation of images to the representation of surfaces. The stabilizing functionals used so far in early vision problems cannot deal adequately with discontinuities, since they offer no control over smoothness. Our recent work by Terzopoulos, Marroquin and Poggio is aimed at a formal treatment of discontinuities as an extension of standard regularization analysis in early vision.

7.1. Discontinuities and splines under tension

Terzopoulos is extending his earlier analysis of the discontinuity problem in computing visible-surface representations. He proposed as a physical model of visual surfaces the "thin plate surface under tension," a natural two-dimensional generalization of the well-known spline under tension. Since surface reconstruction is essentially an ill-posed problem (the available visual constraints do not uniquely determine the surface, see Poggio and Torre, 1984) this class of surfaces can be viewed as defining a stabilizing functional with smoothness properties that can be controlled appropriately at depth and orientation discontinuities. Details and generalizations to other ill-posed vision problems are currently under analysis.

7.2. Beyond Standard Regularization Methods

A related approach to the same problem is being followed by Marroquin and Poggio (Marroquin, 1984), and is based on the work by Geeman and Geeman (1984). The surface to be reconstructed is considered as a sample from a stochastic process based on a Gibbs distribution (that is, it is modelled as a Markov Random Field).

The reconstruction problem is then equivalent to that of finding the best (Bayesian) estimate for the surface, given the information provided by the observations (which themselves may be the output of some process, such as stereo or laser ranging techniques), and the a priori knowledge, both about the properties of the surface (piecewise smoothness, for example), and about the geometry of the discontinuities, if they are present. It is interesting to note that this construction leads to a mathematical formulation which is very similar to the ones obtained from the standard regularization techniques. The best estimate is chosen as the one that minimizes a certain "energy" function that consists of the sum of a term that corresponds to the agreement of the estimate with the observations, and terms

corresponding to the constraints imposed by the prior knowledge about the nature of the solution. This should not come as a surprise, since Bayesian estimation is a regularization method; the functionals obtained by this approach, are, in general, non-convex, and the computational problem of finding their global minimum is, therefore, a complex one.

At present, we have applied this technique to the problem of reconstructing a piecewise smooth surface from sparse and noisy data, and we have obtained encouraging results (using synthetic data). The algorithm simultaneously detects the discontinuities and interpolates smooth surfaces across the appropriate regions. For dense data, the same algorithm may be used for edge detection and image segmentation tasks.

The problem of global minimization of the corresponding energy functionals, is currently being solved using the method of *simulated annealing*, a stochastic technique recently developed by Kirkpatrick et al. (1982), which is effective, but computationally expensive (at least on a serial machine). Currently, research is progressing, aimed at trying to find algorithms that are computationally more efficient, and extending the range of applications of this approach.

8. Shape Representation and Object Recognition

At a higher level we are actively developing several approaches to shape representation and object recognition. Though the primary interest is in the field of robotics, we expect that our research in this area will have a significant fall-out for image understanding in general. We will first describe the approach of Brady and coworkers to the problem of computing, representing and exploiting 2-D and 3-D shapes. We will then outline a system developed for object recognition in robotics by Brou. We will conclude with an algorithm for recognizing polyedral objects from sparse sensory data, due to Grimson and Lozano-Perez.

8.1. Smoothed Local Symmetries, the Curvature Primal Sketch and Reasoning About Shapes

Brady and his colleagues have investigated the representation of two and three dimensional shape. Smoothed local symmetries (Brady and Asada 1984; Brady 1984) represent both the bounding contour and region subtended by a two-dimensional shape. Brady and Asada (1984) have developed a mathematical analysis of the smoothed local symmetry representation, and constructed an efficient implementation. Extensive experiments have been carried out on images of tools, leaves, and animals to determine the stability and sensitivity of the representation. Brady and Asada's implementation performs the following operations: first, the significant changes of curvature are found at a variety of scales; second, the contour is approximated between successive curvature changes by quadratics, and the spines are computed between pairs of quadratics. The spines are displayed as curves; but they are in fact represented internally as descriptions of the parameters proposed by the mathematical analysis.

The multiscale representation of significant curvature changes is called the Curvature Primal Sketch (Asada and Brady 1984). Curvature changes are detected, localized, and assigned a symbolic description by the multiscale patterns of curvature and curvature change peaks to the idealized responses of a set of models that includes corner, crank, and smooth join. The models are analogous to those investigated by Marr (1976) in the original Primal Sketch.

Heide (1984) has based an alternative implementation of smoothed local symmetries on an efficient algorithm developed by Bookstein for the symmetric axis transform. He has developed a hierarchy of symbolic descriptions of a shape that has the raw parameter values of the contour and region at the lowest level, and a symbolic description of the major spines and contours at the top level. The major spines are found by smoothly extending spines whose descriptions are sufficiently similar, and by subsuming spines whose covers are wholly

contained within the cover of some other spine. An intermediate level provides a symbolic description of all the primitive spines. Heide's representation has broad scope. It reduces to that proposed by Hollerbach (1975) for globally symmetric shapes that have no attached subparts.

Although there are many geometrically plausible descriptions of any given shape, the human visual system is usually quite definite about the one that is perceived. Indeed, ambiguity often has to be pointed out for us to realise that it is possible. For example, a square that has a small square removed from one of its corners is rarely perceived as an L-shape, or vice versa. Bagley has investigated this problem for polyhedral shapes using smoothed local symmetries and a database of models. His program generates descriptions that accord well with human perception. Metric information is often important for choosing between alternative descriptions of a shape. Bagley's program is also capable of describing overlapping shapes.

The various implementations of smoothed local symmetries result in a semantic-network-like symbolic description of a shape. We have begun to investigate the usefulness of this representation for perceptual goals other than inspection and recognition. In particular, we have interfaced smoothed local symmetries to Winston's ANALOGY program (Brady, Agre, Braunegg, and Connell, 1984). The resulting program can be taught to recognise simple tool shapes. It can learn to recognise that a tool is a hammer yet learn that there is a functional hierarchy of hammers. Our goal is to be able to reason about objects by relating their shape to their function (see (Winston, Binford, Katz, and Lowry, 1984)). Suppose one's goal is to drive fine tacks into upholstery. We know (somehow) that it is a bad idea to use a typical framing hammer for the job because it tends to break the tacks and destroy the furniture. A specialized tool called a tack hammer has been invented; but one might not be available. Based on a suitable model of hammering and of tool shapes, one learns the trick of grasping a screwdriver by its blade and using the handle to drive the tack. Our program is almost capable of such reasoning. We propose that higher order geometric structures based on smoothed local symmetries directly support reasoning between structure and function.

We have continued to develop a representation of three-dimensional surfaces (Brady and Yuille 1984; Brady, Ponce, Yuille, and Asada 1985). The work has both a theoretical and an empirical component. The theoretical component is a study of classes of surface curves as a source of constraint on the surface on which they lie, and as a basis for describing it. Brady, Ponce, Yuille, and Asada (1985) analyze bounding contours, surface intersections, lines of curvature, and asymptotes. They develop a novel proof of, and extension to, a recent result due to Koenderinck that shows that the sign of the Gaussian curvature of the surface at points along the boundary curve is the same as the sign of the curvature of the projection of the boundary curve.

They also prove a theorem about generalized cones that relates surface curves to a volumetric representation proposed by Marr (1977). Marr considered generalized cones (Brooks 1981; Brooks and Binford, 1980) with straight axes. He suggested that such a generalized cone is effectively represented by (i) those cross-sections, called *skeletons*, for which the expansion function attains an extreme value; and (ii) the tracings, called *flutings*, for which the cross-section function attains an extremum. (A tracing is the space curve formed by a point of the cross-section contour as the cross-section is drawn along the axis.) Brady, Ponce, Yuille, and Asada (1985) show that if the axis of a generalized cone is planar, and the eccentricity of the cone is zero, then (i) a cross section is a line of curvature if and only if the cross section is a skeleton; and (ii) a tracing is a line of curvature if the generalized cone is a tube surface (the expansion function is a constant), or the tracing is a fluting.

The experimental work investigates whether the information suggested by the theoretical analysis can be computed reliably and efficiently. Brady, Ponce, Yuille, and Asada, (1985) demonstrate algorithms

that compute lines of curvature of a (Gaussian smoothed) surface; determine planar patches and umbilic regions; extract axes of surfaces of revolution and tube surfaces. They report preliminary results on adapting the curvature primal sketch algorithms of Asada and Brady (1984) to detect and describe surface intersections.

8.2. Object Representation with EGI

We have used the Extended Gaussian Image (EGI) to represent objects and obtain their orientation in depth maps (Brou, 1983). This approach makes it possible to separate the translation and rotational components of the object position. The EGI is formed by mapping surface information onto a unit sphere. The value assigned to each of the vectors \vec{n} on the sphere is equal to the surface area of the object with normal \vec{n} . Minkowski's theorem position guarantees that this representation is unique for each convex object. Unfortunately ambiguities arise when the object is non-convex but, in any given application, it is unlikely that two non-convex objects will have the same EGI. A CAD system based on constructive solid geometry was built to describe objects to the machine (Brou, 1981). The program obtains the polyhedral representation of the object and constructs the EGI automatically.

The orientation of the object can be obtained from the depth map by forming an EGI of the depth data and comparing it with the EGI of the objects.

Many implementation problems had to be addressed before the successful implementation of the comparator. The first of these is the internal representation of the EGI. Unlike an image that is conveniently stored in a two-dimensional array, the problem is now one of sampling and storing data obtained on a sphere. A geodesic representation derived for the icosahedron was then selected to tessellate the sphere. It allows arbitrarily fine subdivisions and is almost uniform. A tessellation with 500 triangular patches proved to be optimal. The triangles are projected onto the sphere and the spherical patches formed by them define certain ranges of orientations. An algorithm assigns a value to each of these cells by computing the surface area of the object pointing in the range of direction defined by the cell.

The second important problem is the comparison of the two models. Finding the orientation of the object in the image is equivalent to identifying the relative orientation of the distributions on the two spheres. It is basically a correlation problem, but in the space of rotations. It is then necessary to determine the rotations that will be used to compare the models. This was solved by representing rotations as quaternions and considering them as points on a unit sphere in a four dimensional space. By studying regular figures in that space, an algorithm was developed to uniformly sample the space of rotations with arbitrarily large numbers of points. A final verification was done to guarantee that the grids of the tessellated sphere line up for each of the rotations in the set. Unfortunately this means that multiple EGIs of each object are required (one for each set of 60 rotations).

All the algorithms were tested with real and artificial data. The experiments revealed that the most difficult factor to deal with is the size of the space of rotations. Even when about 6000 rotations are used to compare the two models, errors of 16 degrees are possible. This is acceptable for some pick and place operations, but in order to obtain the high level of accuracy required for assembly, hundreds of thousands of rotations would be required. From this analysis, it was concluded that the EGI technique can still be used to obtain an estimate of the object's orientation. Feature based techniques would then be required to reduce the error range to a few degrees.

8.3. Object Recognition from Sparse Visual Data

An alternate approach by Grimson and Lozano-Perez to object recognition is aimed at using very simple, sparse, potentially noisy sensory measurements. It assumes that sensory information about a scene can be processed to obtain a set of estimates of the position

and surface orientation of small patches on a surface. Because of the simplicity of the sensory requirements, many different sensing modalities could be used to obtain the data. In order to recognize instances of known objects in the scene, the sparse sensory is matched against polyhedral models of the known objects, having up to six degrees of freedom relative to the sensor (three translational and three rotational). We stress that the objects need not be themselves polyhedral, only that they can be so modeled to within some bounded error.

The approach operates by examining hypotheses about pairings between sensed points and object surfaces. Of course, examining all possible assignments of sensed points to object surfaces is infeasible for all but trivial cases. The key to the approach is identifying simple, robust constraints that will effectively and efficiently reduce the size of the portions of the search space that must be explicitly explored. We have found (Grimson and Lozano-Perez 84) that a very effective set of coordinate-frame-independent constraints can be derived by considering local constraints on: (1) distances between faces, (2) angles between face normals, and (3) angles, relative to the surface normals, of vectors between sensed points. These constraints turn out to be very efficient in reducing the number of feasible interpretations of the sensory data, usually to a unique interpretation, modulo partial symmetries of the object.

The constraints have several advantages. They are coordinate-frame-independent, so that the object is recognized independent of any peculiarities of the sensor's coordinate system. In other words, the objects are recognized by matching intrinsic shape characteristics. Only after all feasible interpretations are found, does the algorithm explicitly solve for a legal transformation from model coordinates to sensor coordinates, thereby localizing the object. The constraints also demonstrate a strong degree of robustness to noise, degrading gracefully as the sensor measurements are increasingly perturbed. We have routinely run the algorithm successfully with errors in measuring surface orientation on the order of 30° and with errors in measuring surface position of 1 part in 50. The constraints also straightforwardly extend to the case of recognizing overlapping or partially occluded objects. As well, the constraints apply both to the simple case of isolated objects in stable positions (three degrees of freedom) and to the more complex case of arbitrarily oriented objects (six degrees of freedom). Because of the simplicity of the technique, it has been implemented in a table-lookup algorithm that is quite fast.

The algorithm has been tested on both synthetic and real data. We have run extensive simulations on both two dimensional and three dimensional objects, under a variety of situations, including widely varying ranges of simulated error. The success of these simulations in identifying objects in the presence of noise from very few data points has been further supported by a theoretical combinatorial analysis (Grimson 84). We have also successfully tested versions of the algorithm on real data obtained from grey level image data and laser ranging devices. M. Drumheller has applied it to noisy and sparse sonar sensor data.

9. The Analysis of Spatial Relations

Today, there is still a notable lack of solid methods for analyzing shapes and spatial relations in images. For example, there are few mechanisms for dealing with questions about spatial relations such as "does object A lie within object B," "is A supported from below," "can A fit in the space between B and C." The ability to process spatial information efficiently and provide answers to questions regarding shape properties and spatial relations is crucial for the tasks of object recognition, visually guided manipulation, and reasoning about scenes.

In addition to the lack of suitable algorithms, standard computer architecture is inappropriate for this important task, and this deficiency

hinders the interpretation and use of visual information.

The problem of computing general shape properties and spatial relations among objects and object parts is a novel domain of investigation. Mahoney and Ullman have begun a study of the analysis of spatial information and development of new algorithms for the computation of spatial relations. Mahoney has chosen as a specific domain for this project the problem of interpreting terrain maps, since this requires the computation of many simple and often quite difficult spatial relations from rather simple primitives, mainly line drawings. We expect that for achieving the required level of performance in this task the use of parallel operations would be required. This will require in turn the use of new types of processors, specialized for the parallel processing of spatial information.

Computation of simple spatial relations is very often global, as is well known from the Perceptons work of Minsky and Papert. We discuss the problem of computing spatial relations at some length in the following subsections, because we believe it is a critical issue for attacking the problems of intermediate and high level vision. They suggest a class of algorithms that are not of the regularizing type and which require a hardware facility in a parallel architecture — such as the Connection Machine router — to support the use of pointers and non-local connections.

9.1. Terrain Maps and the Study of Spatial Analysis

The use of diagrams is remarkably effective in a wide range of human problem-solving situations. Diagrams provide a rich, compact external information store, from which visual processes rapidly extract just the information relevant to the task at hand. Moreover, the fact that people often find it useful to *draw* diagrams in the course of solving a problem suggests that visual processing can play an integral part in reasoning.

The use of terrain maps — applied to navigation, for example — provides a very *general* example of this phenomenon, in the sense that maps pose most of the visual problems found in a range of other schematic representations. Notice that navigation might involve not only examining a map, but also *sketching* selected aspects of it, or *registering* one map with another.

Terrain maps represent the physical elements of the landscape and their spatial distribution. The *descriptive elements* of a map include primarily plane curves and symbols. Curves are used to represent both linear geographic features — such as rivers and roads — and the *boundaries* of areal geographic features — such as landmasses and political boundaries. Symbols — cartographic and alphanumeric characters — are used as place markers for geographic features that have little or no spatial extent at the scale of the map (like summits), or they are used to *label* other items.

Even simple use of a map requires the following capabilities:

1. Distinguish and classify *individual* descriptive elements. For example, a given line in the map must be interpreted as a river, road, shoreline, etc. (This is a problem in segmentation/grouping and recognition).
2. Interpret *combinations* of descriptive elements as representations of geographic items. For example, a set of closely spaced contour lines that are nearly circular and concentric might represent a volcano. (This, also, is a problem in grouping and recognition.)
3. Interpret references to locations in the map. There are several different coordinate systems defined in a map in which such references might be expressed. "North-west", "top-left", "P-16", "Latitude 4, longitude 4", and "central Africa" are instances of coordinates in different map-based reference frames, and whose interpretation depends on sophisticated spatial analysis.
4. Search for specified markers, descriptive elements, or combinations of them. Every form of map use involves, as an initial step, finding "it" on the map, where "it" can be an arrow, a name, or a *pattern* of lines with specified properties — like a volcano. The utilitarian nature of maps makes visual search a very prominent

spatial analysis operation. The rich context present in a map makes the search problem non-trivial, especially when human performance at such tasks is taken as a standard for efficiency.

5. Determine spatial properties of the items in the map, or relations holding between them. The interesting properties and relations in a map are those which have useful correlates in the geographical world. Properties of and relations between landscape features can map directly or indirectly to two-dimensional properties and relations in the map. The length of a river is a case of the direct case; the elevation of a town must be inferred from the nesting of the marker for the town within the contour lines.

Typical map use can be characterised in terms of two interacting systems. Above, there is a reasoning system engaged in a task which involves knowledge of the geographical world. At its disposal is a *partial*, declarative model of the world, from which certain inferences can be made. Below, there is a spatial analysis system whose input is an image of a map, and which is capable of resolving spatial queries given by the reasoning system, through the coordinated application of the capabilities described above. The answers to these queries are used to extend or correct the high-level geographical-world model. Our eventual goal is to develop such a spatial analysis system.

The following mock "tour" of an imaginary map illustrates some of the requirements on a spatial analysis system, in terms of a few of the kinds of spatial properties and relations it must discriminate. Such a sequence might be given to a map user on the other end of a radio link.

1. Find the arrow in the upper left section of the map.
2. Find the town pointed at by the arrow.
3. Visually follow the road the town is on to the second river crossing.
4. Visually follow the river to its source.
5. Find the peak of the mountain in which the river has its source.
6. Find Robot State Park, which is due north of this peak.
7. Find the point marked X inside the state park.
8. What is marked by the X?

9.2. A few elemental spatial operations go a long way

The *visual routines* paradigm, introduced by Ullman, proposes that an open-ended set of abstract spatial properties and relations can be computed by integrating basic spatial operations taken from a small fixed set into specialized procedures. Being the basis of most other spatial analysis, these elemental operations must be simple, general, very robust, and very fast.

We have begun to study visual tasks in the context of schematic drawings, with a view to the following issues:

1. What are the objects of spatial analysis, and what are the important spatial relations and properties they exhibit.
2. What are robust, efficient visual routines for computing these objects, properties, and relations; from what basic operations are these routines composed.

The standards for efficiency and robustness come from *human* performance at similar tasks.

3. How should the processing be *controlled* such that, in a real visual problem solving situation, the appropriate routines are applied productively.
4. What sort of architecture will effectively support the suggested basic operations and control structure.

Ullman has made a number of suggestions for generally useful basic operations, justified primarily on computational grounds. These operations include contour tracking, area coloring, ray projection, marking locations for later processing, shifting the processing focus, and indexing to locations which have salient properties.

We have begun studies of visual tasks of the sort that arise in maps and diagrams, in order to further define, evaluate, and complete this partial set of basic operations and the processing model of which they are a part. Initial efforts indicate that the operations listed above can account, in large part, for a wide variety of the needed spatial relations.

Current efforts are aimed at (a) determining what additional basic operations are required and (b) designing an efficient implementation for the basic operations that are being used.

9.3. Pointers are useful for spatial analysis.

We have begun to consider the possible implementations of spatial operations on the Connection Machine. The following subsections contain initial proposals for the use of non-local interconnections in the course of spatial analysis.

9.3.1. Directing Processing To Salient Locations

An important operation in spatial analysis is the intelligent selection of locations for initial processing. Locations that are *prominent* in some (selected) property or feature—such as brightness contrast, relative motion, or the presence of line terminations, crossings, or curvature changes—are often good candidates. We believe it is important that vision hardware be able to test for prominence in the property of interest at all locations simultaneously; then further processing can be selectively applied to those locations at which the test is successful.

So-called winner-take-all mechanisms have been proposed to account for how the most prominent location might be chosen among all of those having a given property (Koch & Ullman, 1984). One way of accomplishing this with non-local communication is to determine the location with the maximum value by an iterative procedure much like an auction. When all but the most prominent location has dropped out of the bidding, the processor representing this location sends its address to the most-prominent-location cell preassigned for the given property. Processing of this location may now proceed by making use of the address this special cell contains (Koch & Ullman, 1984).

9.3.2. Concurrent Computation of Spatial Relations

On machines with non-local communication, spatial properties or relations may be computed concurrently in a single application of the operation if the relevant scene items have been uniquely identified first.

For example, all the inside/outside relations occurring among some set of boundaries and figures could be computed at once by uniquely coloring the interiors of all boundaries (which colors the figures they contain), and then collecting the addresses of all figures of each color.

9.3.3. Linking Symbolic and Iconic Visual Representations

As spatial analysis is incrementally applied to the early, more iconic representations, more symbolic representations of portions (or aspects) of the scene are produced. It is useful to link these symbolic and iconic components, for each scene item, using pointers. For example, through such pointers, operations at the symbolic level can initiate further image level processing when necessary.

9.3.4. Optimal Algorithms for Elemental Spatial Operations

Coloring (or region growing) is one important class of elemental operations. Using something like the Connection Machine's fixed NEWS connection network, coloring can be implemented with run time proportional to the diameter of the region. On a serial architecture the time is proportional to the region's area.

However, this is only a lower bound on the performance we seek: observations of human vision suggest that the corresponding operations take roughly constant time—up to quite large diameters. We believe that such performance will be necessary for full-scale, real-time vision.

Achieving the required performance for these operations will involve discovering novel parallel computation structures. Such observations have architectural consequences for experimental systems: using the Connection Machine's programmable interconnect will provide an experimental medium for this study.

For example, coloring can be speeded up dramatically—in terms of the number of algorithmic steps it takes—when the non-local communication is used appropriately. This is for the same reason that a figure can be painted more quickly with a large brush than a small one; the analogue of brush size on machines with non-local communication is the size of the pixel neighborhoods that are turned on at each step. Again such observations have architectural consequences for experimental systems: using the Connection Machine's programmable interconnect will enable us to experiment with various brush sizes.

Systematic and extensive simulations can lead to an understanding of how the optimal size and shape of the brush, in this example, are constrained by particular kinds of input figures. (A tree might require different strategies for fast coloring than the disc of the moon.) The advantage of the Connection Machine as a simulation tool comes not only from speed—programmable interconnections provide a substantial conceptual advantage in the design and simulation of novel parallel algorithms. Similar advantage has been experienced in the use of high-level languages, or in the gradual adoption of what were once strictly language-level concepts in the design of modern conventional architectures. Naturally, the benefits will be even greater as effective programming-language concepts are developed.

10. Conclusions

In this report we have reviewed some of our work over the last year on Vision and Image Understanding. Our effort spans various levels. It begins with the early problems of computing surface distances and other surface properties (the $2\frac{1}{2}$ -D sketch). We have analyzed in depth the problem of edge detection, multiple scales, stereo and motion. We have implemented algorithms for solving these problems efficiently. We now plan to refine our analysis of the individual modules of early vision, consider new ones and attack the problem of their fusion for a reliable and robust computation of the $2\frac{1}{2}$ -D sketch. At a higher level we are studying the problem of representing shape and recognizing objects from different, complementary perspectives. At a still higher level we have described in some detail the idea of *visual routines*, as a key area of investigation for solving efficiently and reliably many high level visual tasks, including navigation and recognition.

The new idea of *regularization analysis* promises to unify at least part of our research in early vision. It also suggests a preliminary classification of parallel architectures for vision. Regularization algorithms of early vision typically process retinotopic arrays of data with only local connections. Non-regularization algorithms, such as for instance some of the visual routines, are best implemented with hardware facilities capable of manipulating pointers and setting up virtual connections between spatially non-adjacent processors. The Connection Machine, being developed at the Artificial Intelligence Laboratory, promises to become soon a fertile ground for experimenting with efficient parallel implementations of our vision algorithms.

References

- Asada, Haruo and Michael Brady, [1984]. The curvature primal sketch, MIT Artificial Intelligence Laboratory, AIM-758.
- Bagley, S. [1984 (in preparation)]. Model-based representation of two-dimensional shapes with intrusions and extrusions, MIT Artificial Intelligence Laboratory.
- Baker, H. H., [September, 1982]. Depth from edge and intensity based stereo., Stanford University Technical Report STAN-CS-82-930.
- Brady, Michael, [1983b]. Criteria for shape representations, *Human and Machine vision*, Beck J. and Rosenfeld A., eds., Academic Press.
- Brady, Michael, [1984a]. Representing shape, *Proc. IEEE Conference on Robotics, Atlanta, Ga.*
- Brady, Michael, [1984b]. "Artificial Intelligence and Robotics," *Artificial Intelligence*.
- Brady, Michael, and Asada, Haruo, [Fall 1984]. "Smoothed local symmetries and their implementation," *Int. J. Robotics Research*, 3 (3).
- Brady, Michael, and Yuille, Alan, [1984a]. "An extremum principle for shape from contour," *IEEE PAMI*, 6(3), 288 - 301.
- Brady, Michael, and Yuille, Alan, [1984b]. Representing three-dimensional shape, Romansy Conf., Udine, Italy.
- Brady, Michael, Agre Philip, Braunneg David J., and Connell Jonathan H., [1984]. The Mechanic's Mate, The European Conference on Artificial Intelligence, Pisa, Italy.
- Brady, Michael, Ponce, Jean, Yuille, Alan, and Asada, Haruo, [1985]. Describing surfaces, *Proceedings of the 2nd Int. Symp. Rob. Res.* Hanafusa H., and Inoue, H., (eds.), MIT Press.
- Brou, P., [1981]. Implementation of high level commands for robots, Master of Science thesis, Department of Electrical Engineering and Computer Science (M.I.T.).
- Brou, P., [1983]. Finding the orientation of objects in vector maps, Ph.D., Electrical Engineering and Computer Science (M.I.T.).
- Babaud, J., Witkin, A. and Duda, R., [1983]. Uniqueness of the gaussian kernel for scale-space filtering, Fairchild TR 645, Flair 22.
- Baker, H. H. and Binford, T. O., [August 1981, 631-636]. Depth from Edge and Intensity Based Stereo., *Seventh International Joint Conference on Artificial Intelligence*.
- Canny, J. F., [1983]. Finding edges and lines, MIT Technical Report No. 720.
- Davis, L., [1975]. "A survey of edge detection techniques," *Computer Graphics and Image Processing*, 4, 248-270.
- Crowley, J.L., [1982]. A representation for visual information, CMU-RI-TR-82-7, Robotics Institute Carnegie-Mellon University.
- Genian, S. and Geman, D., [1983]. Stochastic Relaxation, Gibbs Distributions, and the Bayesian restoration of images., Brown University.
- Grimson, W.F.L., [1981]. *From Images to surfaces*, MIT Press, Cambridge, Mass.
- Grimson, W.F.L., [1983]. "Surface consistency constraints in vision," *Comp. C.V.G.I.P.*, 24, 28-51.
- Grimson, W. F. L., [1984]. Computational Experiments with a Feature Based Stereo Algorithm, A. I. Memo 762, Mass. Institute of Technology.
- Grimson, W. F. L. and Lozano-Pérez, T., [1984]. Model-Based Recognition and Localization from Sparse Range or Tactile Data. To appear in *International Journal of Robotics Research*, (Also appeared as MIT AI Lab Memo 738.).
- Grimson, W. F. L., [1984]. The combinatorics of local constraints in Model-Based Recognition and Localization from Sparse Data., MIT AI Lab Memo 763.
- Hackbusch, W., and Trottenberg, U., (eds.), [1982]. *Multigrid Methods, Lecture Notes in Mathematics*, Vol. 960, Springer-Verlag, New York.
- Haralick, R.M., [1982]. Zero-crossings of second directional derivative edge operator, SPIE Proc. on Robot Vision, Arlington Virginia.
- Heide, S., [1984]. A hierarchical representation of shape from smoothed local symmetries, MIT Artificial Intelligence Laboratory.
- Kirkpatrick, S., Gelatt, C.D., & Vecchi, M.P., [1983]. "Optimization by Simulated Annealing," *Science*, 220, 671-680.
- Koch, C., & Ullman, S., [1984]. Selecting one among the many: a simple network implementing shifts in selective visual attention, A. I. Memo 770, Mass. Institute of Technology.
- Logan, B.F., [1977]. "Information in the Zero Crossings of Bandpass Signals," *Bell Sys. Tech. J.*, 56, 4, 487-510.
- Mayhew, J. F. W., [1982]. "Stereopsis," in: *Physical and Biological Processing of Images*, O. J. Braddick and A. C. Sleight, eds., Springer-Verlag, Berlin.
- Mayhew, J.F.W. and Frisby, J.P., [1981]. "Psychophysical and computational studies towards a theory of human stereopsis," *Artificial Intelligence*, 17, 349-385.
- Mayhew, J.F.W. and Longuet-Higgins, H.C., [1982]. "A computational iyp model of binocular depth perception," *Nature Lond.*, 297, 376-379.
- Marr, David, [1982]. *Vision: A computational investigation into the human representation & processing of visual information*, W. H. Freeman & Co., San Francisco.
- Marr, D. & Hildreth, E., [1980]. "Theory of Edge Detection," *Proc. R. Soc. Lond. B*, 207, 187-217.
- Marr, D., Poggio, T., [1979]. "A computational theory of human stereo vision," *Proc. R. Soc. Lond. B*, 204, 301-328.
- Marr, D., Poggio, T., Ullman, S., [1979]. "Bandpass channels, zero-crossings and early visual information processing," *J. Opt. Soc.*, 70, 868-870.
- Marroquin, J.L., [1984]. Boundary preserving smoothing and interpolation, MIT Artificial Intelligence Laboratory, AIM-792.
- Nishihara, H.K., [1981]. "Intensity, visible-surface, and volumetric representations," *Artificial Intelligence*, 17, 265-284.
- Nishihara, H.K., [1984]. PRISM: A practical real-Time Stereo Matcher, MIT Artificial Intelligence Laboratory, AIM-780.
- Nishihara, H. K. and Poggio, T., [1984]. Stereo vision for robotics, Proceedings of the First International Symposium of Robotics Research.
- Ohta, Y. and Kanade, T., [1983]. Stereo by intra- and inter-scanline search using dynamic programming, Carnegie-Mellon University Technical Report CMU-CS-83-162.
- Poggio, G. and Poggio, T., [1984]. "The analysis of stereopsis," *Annual Review of Neuroscience*, 7, 379-412.
- Poggio, T., [1984]. "Vision by Man and Machine," *Scientific American*, 250, 4, 106-115.
- Poggio, T., [1984]. "Routing Thoughts," *Artificial Intelligence Laboratory Working Paper*, 258.
- Poggio, T. and Torre, V., [April, 1984]. "Ill-posed problems and regularization analysis in early vision," *Center for Biological Information Processing Memo*, 001.

- Poggio, T. and Koch, C., [May, 1984], "An analog model of computation for the ill-posed problems of early vision," *Center for Biological Information Processing Memo*, 002.
- Poggio, T., Voorhees, H. and Yuille, A., [1984], "A regularized solution to edge detection", in preparation.
- Poggio, T., Torre, V., [1984], "Ill-Posed Problems and Variational Principles in Vision," *A. I. Memo*, 773.
- Rosenfeld, A., [1982], Quadrees and Pyramids: hierarchical representation of images, TR 1171, University of Maryland.
- Rosenfeld, A. and Kak, A. C., [1982], *Digital Picture Processing*, Academic Press, New York.
- Stansfield, J. L., [1980], "Conclusions from the commodity expert project," *MIT Artificial Intelligence Lab Memo*, 601.
- Terzopoulos, D., Multilevel reconstruction of visual surfaces: Variational principles and finite element representations, MIT A.I. Lab., Cambridge, MA, 1982, AI Memo No. 671, reprinted in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (ed.), Springer-Verlag, New York, 1984, 237-310.
- Terzopoulos, D., [1983a], "Multilevel computational processes for visual surface reconstruction," *Computer Vision, Graphics, and Image Processing*, 24, 52-96.
- Terzopoulos, D., [August, 1983b, 1073-1077], The role of constraints and discontinuities in visible-surface reconstruction, Proc. Eighth Inter. Joint Conf. Artificial Intelligence (IJCAI-83) Karlsruhe, W. Germany.
- Terzopoulos, D., [January, 1984a], Multiresolution Computation of Visible-Surface Representations, Ph.D., Dept. of Electrical Engineering and Computer Science, MIT, Cambridge, MA.
- Terzopoulos, D., [1984b], "Integrating visual information from multiple sources for the cooperative computation of surface shape," *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*, A.P. Pentland (ed.), Ablex, New Jersey.
- Terzopoulos, D., [1984c], "Multiresolution algorithms in computational vision," *Image Understanding 1984*, S. Ullman and W. Richards (eds.), New Jersey.
- Terzopoulos, D., [August, 1984d, 314-317], Efficient multiresolution algorithms for computing lightness, shape-from-shading, and optical flow, Proc. 1984 National Conf. on Artificial Intelligence (AAAI-84), Austin, Texas.
- Tikhonov, A. N., Arsenin, V. Y., [1977], *Solution of Ill-Posed Problems*, Winston and Wiley Publishers, Washington.
- Torre, V. and Poggio, T., [1984], "On Edge Detection," *MIT A.I. Memo*, 768.
- Verri, A., [1984], *Methodi Matematici per la Visione Stereografica*, Ph.D., University of Genoa.
- Witkin, A., [1983], Scale-Space Filtering, Proceedings of IJCAI, 1019-1021, Karlsruhe.
- Yuille, A.L., [1983], "The smoothest velocity field and token matching schemes," *MIT A.I. Memo*, 724.
- Yuille, A.L. and Poggio, T., [June, 1983a], "Scaling Theorems for Zero-crossings," *MIT A.I. Memo*, 722.
- Yuille, A.L. and Poggio, T., [October, 1983b], "Fingerprints Theorems for Zero-crossings," *MIT A.I. Memo*, 730.
- Yuille, A.L. and Poggio, T., [in preparation, 1984a], "Fingerprints and the Psychophysics of Gratings," *MIT A.I. Memo*, 751.
- Yuille, A.L. and Poggio, T., [in preparation, 1984b], "Fingerprints and their slope," *MIT A.I. Memo*, 752.
- Yuille, A.L. and Poggio, T., [in preparation, 1984b], "A generalized ordering constraint for stereo correspondence," *MIT A.I. Memo*, 777.

SPATIAL UNDERSTANDING

Thomas O. Binford

*Artificial Intelligence Laboratory
Stanford University, CA 94305*

Abstract

We describe research in intelligent systems for Image Understanding. The ACRONYM system has been used in recognition of industrial parts in the Intelligent Task Automation project. A system for intelligent matching in stereo and motion sequences is under development. A geometric modeling system using stereo images has been implemented. A sophisticated symbolic graphics system for a very general class of generalized cylinders is nearing completion. A preliminary report is presented of reformulation of problems in computational geometry. Continuing analysis of the interpretation of line drawings is described. New results have been obtained in aggregation and in edge segmentation. A survey is presented of commercial array processors.

I. Introduction

This report describes research in intelligent systems for Image Understanding which includes geometric modeling, geometric reasoning, analysis of image structure, and interpretation of three dimensional structure. One goal of the research is an intelligent stereo mapping system.

[Chelberg 84] describe using the previous intelligent system, ACRONYM [Brooks 84], for recognition of industrial parts in the Intelligent Task Automation project. Several extensions were necessary in geometric modeling and geometric reasoning, extensions which carry over to the SUCCESSOR system. The experience in this project has sharpened requirements for the new system.

A system for obtaining correspondence in stereo and motion sequences is near the stage of matching image sequences [Dreschler-Fischer 84]. The system uses curves and corners as image features, however it performs structural matching. It groups features within single images into constellations and generates plans for data-driven matching.

A geometric modeling system has been built which enables construction of generalized cylinder models and which determines a class structure of models [Takamura 84]. The user interacts with the system by menu selection using a voice input device, using the keyboard only for

naming parts. Models of objects are built up from stereo pictures of objects by using a pointing device with a stereo station. The system minimizes the number of input points required for defining generalized cylinder parts.

A system is nearly completed for symbolic graphic output for a very arge class of generalized cylinders [Scott 84]. Generalized cylinders are specified by spine, cross section, and sweeping rule, each of which can be a general function of one variable for this system. The system is novel in the large class of objects which can be displayed. The hidden surface algorithm is related to ray tracing which is the method of choice in high performance graphics. Although ray tracing is a brute force procedure which consumes great computational resources, this research includes an analysis of computational complexity and has resulted in conceptual improvements to limit complexity. This research has led to some concepts for generating generic, symbolic predictions of possible views of objects.

Research into choosing representations for solving problems in geometric reasoning led into specializing in reformulating problems in computational geometry [Lowry 84]. Computational geometry algorithms have special relevance in geometric reasoning and geometric database operations. This research deals with problem-solving in computational geometry with computational complexity considerations foremost, related to strategy selection for efficient algorithms in perceptual reasoning. In other research in geometric reasoning, Chelberg (unpublished) made a system for symbolic manipulation for problem simplification by approximation of algebraic constraints.

[Malik 84] describes inference of surfaces from line drawings. Inference rules and geometric reasoning are described together with an enlightening example. The approach is particularly relevant because previous analyses had ambiguities of 100 to 200 for the simplest drawings, that of a tetrahedron.

Research continues on aggregation and segmentation. [Lowe 84] describes a program which groups curve segments which form continuous curves. [Nalwa 84] describes a directional edge operator. [Triendl 84] describes an edge finding system which determines extended curves with cir-

cular arc and straight segments. Dreschler has implemented her corner finder.

An evaluation of short term improvements to computation power for IU has been carried out in the form of a survey of commercially available array processors [Lim 84].

[Blicher 84] notes that dimensional arguments dictate that edges cannot be localized simultaneously in angle and transverse position by zero crossings of a single convolution operator.

A mobile robot is on the verge of being operational without sensing. It has an onboard 68000 and LSI/11, digital communication at 1200 baud, analog tv transmitter and Polaroid acoustic sensors.

II Systems

The Intelligent Task Automation project dealt with location and assembly of a tray of fifteen parts in an uncontrolled environment [Chelberg 84]. Occlusion and moderate leaning of parts on one another were permitted. This was a new class of parts for ACRONYM. All parts were modeled. To do so required extending the modeling system to include holes or negative volumes, helices, and to represent stable states. ACRONYM was augmented to include: stable states in reasoning; to reason about holes; to predict new relations among primitives, specifically concentric relations between concentric cylinders translated into predictions about concentric ellipses; to predict connected relations between ellipses and ribbons; to predict parallel relations between coils of a spring, and enclosed relations between coils of a spring and an enclosing boundary of the spring. While some code to predict ellipses had been in the original ACRONYM code, ellipse prediction and matching were not fully implemented.

Of the fifteen parts, ten have been recognized in any stable state. Predictions were generated automatically for the remaining five parts but matching has not been achieved for them.

[Dreschler-Fischer 84] describes a system for correspondence in stereo pairs and motion sequences. Constraints in correspondence are maintained as separate knowledge

sources which can change dynamically according to knowledge acquired in operation. Corners and curves are the features used in the system. However, it analyzes structures or constellations of these features in each image and plans a matching sequence which seems effective for the image. It classifies corners formed by curves, especially to relax constraints on T-junctions which indicate occlusion and for which no correspondence is expected. The system groups features into similarity classes which might be ambiguous under the local matching operation. For example, for a checker board, interior corners are all similar, while the four corners of the checker board are unique. The system begins by matching them. Corners connected to them are then unique. Matching takes place between classes of features, rather than individual features. Matching has been tested only on artificial data.

III Geometric Modeling

The geometric modeling system [Takamura 84] has a user interface which relies on commands invoked from menus and selected with voice input. The keyboard is used for naming elements. Geometric forms are specified by points which are entered by pointing devices, in this case a trackball. Points are three dimensional points determined from stereo pairs of pictures. Generalized cylinders are specified by cross section, spine, and sweeping rule. Cross sections can be specified by a few points, e.g. three points for a circle or rectangle. For the class of generalized cylinders with straight spine and constant sweeping rule, only two cross sections are required. For some simple cross sections, a total of only four points are required for an object. For complex parts, still relatively few points are required, of order ten points. Parts can be defined from others by symmetry. A model of a simple object with a few parts can be built in 10 minutes. The system writes out a textual model of the part. It also determines object classes by generalization of constraints which determine object classes in ACRONYM.

A symbolic display system generates a projection of visible surfaces of a very general subclass of generalized cylinders [Scott 84]. Conceptually it can be thought of as ray tracing, i.e. projecting back rays from each pixel to intersect objects, ordering intersections of surfaces along each ray by distance from the image. However, it is waste-

ful to order surfaces along each ray, since order relations change only along boundaries, a one-dimensional subset of rays. In fact, depth relations change only at T vertices and cusps, a zero-dimensional subset of rays. This is a striking decrease in computation in depth ordering.

Limbs of generalized cylinders are obtained by an iterative search stepwise along the surface. Surfaces are specified by spine, cross section, and sweeping rule, each of which may be an arbitrary function of one parameter which is evaluated at each step. Steps are chosen according to a uniform quality criterion. The step distance was chosen to give a constant number of iterations per step. About 100 steps were required for a three turn helix. Objects are defined by unions, intersections, and negatives of these primitive volumes or surfaces.

Projections of limbs are put into an image quad-tree from which surface ordering is obtained. T-junctions are found here. Because the determination of limbs is approximate stepwise, some degradation can occur from lack of resolution. An algorithm has not been yet worked out to obtain consistent, realizable approximations, but an outline is given for a solution.

The system has generated hidden surface views of complex parts, but the full system for structured objects is still being implemented.

From this research has come the problem of determining the locus of points at which the qualitative structure of projections changes. This may lead to compact prediction of image appearance for complex objects.

IV Geometric Reasoning

[Lowry 84] describes considerations for design of a system for automatic reformulation of algorithms in computational geometry. An implementation is underway of a system to solve some problems in computational geometry with computationally efficient solutions. The system is intended to use general methods which are not tailored to the particular problems to be solved. Two approaches are distinguished. The first approach seeks to transform the problem into the form of a computational schema like divide and conquer. The second approach seeks to discover properties of the problem which can be exploited

by problem-solving methods. The first is called schema-driven, the second constraint-driven.

[Malik 84] describes extensions of the analysis of [Binford 81] for interpretation of surfaces from line drawings. Implementation has begun of a program to interpret complex line drawings. The initial analysis is primarily aimed at drawings with straight lines. The notion of minimum number of surfaces was introduced, leading to analysis and constraints on invisible surfaces. Coplanarity rules have been introduced to identify surfaces coincident with lines. A particularly interesting example is used to demonstrate these constraints.

The limitations of previous analyses were most striking in that they tried to find all solutions. [Draper 80] shows that there are many solutions for the simplest drawings. The current analysis promises to reduce this ambiguity extensively.

V Segmentation

[Nalwa 84] describes a directional operator for determining edge elements over a small disk. It has subpixel position resolution and 5 degree angle resolution for step to noise of 2. The operator has a one-dimensional function oriented at an angle; its cross-section function is tanh. First, it determines the orientation of the intensity surface in the window by fitting a plane. Then it fits a cubic surface, then a tanh surface, and a quadratic surface fit for comparison to determine presence or absence of an edge.

REFERENCES

- [Binford 81] Binford, T. O.; "Inferring Surfaces from Images"; Artificial Intelligence Journal August, 1981.
- [Blicher 84] A.P.Blicher; "Edge Localization in both theta and x"; Proc Image Understanding Workshop, 1984.
- [Brooks 81] Brooks, R.A.; "Symbolic Reasoning among 3-D Models and 2-D Images"; Artificial Intelligence Journal, August, 1981.
- [Chelberg 84] D. Chelberg, H. Lim, C. Cowan; "ACRONYM Model-Based Vision in the Intelligent Task Automation Project"; Proc Image Understanding Workshop, 1984.

[Dreschler-Fischer 84] L. Dreschler-Fischer and E. Triendl; "The ASTERIX-System: A Feature-Based Approach to the Correspondence Problem"; Proc Image Understanding Workshop, 1984.

[Lim 84] H.S.Lim and T. Binford; "Survey of Array Processors"; Proc Image Understanding Workshop, 1984.

[Lowe 84] D. Lowe; "Perceptual Organization and Visual Recognition"; Computer Science Dept, Stanford University, forthcoming.

[Lowry 84] M. Lowry; "Automatic Reformulation of Algorithms in Computational Geometry"; Proc Image Understanding Workshop, 1984.

[Malik 84] J. Malik and T. Binford; "A Theory of Line Drawing Interpretation"; Proc Image Understanding Workshop, 1984.

[Nalwa 84] V. Nalwa; "On Detecting Edges"; Proc Image Understanding Workshop, 1984.

[Scott 84] R. Scott; "Graphics and Prediction from Models"; Proc Image Understanding Workshop, 1984.

[Takamura 84] J. Takamura and T. Binford; "Stereo Modeling System: A Geometric Modeling System for Modeling Object Instance and Class"; Proc Image Understanding Workshop, 1984.

[Triendl 84] E. Triendl; "The Edge Appearance Model in a Rule-Based System"; forthcoming.

Recent Results of the Rochester Image Understanding Project

J.A. Feldman, D.H. Ballard and C.M. Brown

Computer Science Department
University of Rochester
Rochester, New York 14627

1. Robust Vision Operators

1.1. Parameter Networks and the Hough Transform

One of the most difficult problems in vision is segmentation. Recent work has shown how to calculate intrinsic images (e.g., optical flow, surface orientation, occluding contour, and disparity). These images are distinctly easier to segment than the original intensity images. Such techniques can be greatly improved by incorporating Hough methods. The Hough transform idea has been developed into a general control technique. Intrinsic image points are mapped (many to one) into 'parameter networks' [Ballard, 1983]. This theory explains segmentation in terms of highly parallel cooperative computation among intrinsic images and a set of parameter spaces at different levels of abstraction.

The most recent application of these ideas are to improved shape-from-shading calculations which work on several spaces [Brown et al., 1983] and motion extraction [Ballard & Kimball, 1983]. This domain specific effort is closely linked to our new work on a more general theory of Hough-like computations and general implementation techniques for them.

The theory is also useful in analysis of cache-based Hough Transform implementations. It is an appealing idea to use a small content-addressable store to accumulate Hough transform results, rather than a potentially huge multi-dimensional array. The initial technical issues were discussed in [Brown & Sher, 1982]. More recent developments are presented in [Brown, 1983; Brown, 1984]. We are currently pursuing VLSI implementations.

1.2 Hough Transform Implementation

Earlier work on the Hough transform [Brown, 1983; Brown & Sher, 1982] has led in three directions.

- 1) Research toward a theory of cache accumulator arrays [Loui, 1983; Brown & Feldman, 1983]
- 2) Experiments with complementary HT and cache management strategies [Brown et al., 1983]
- 3) Hardware (VLSI) designs for HT vote caches [Sher & Tevanian, 1983].

Work in each of these directions is in progress; some of the cited references are draft documents. The behavior of caching schemes for accumulation of votes in the Hough transform is equivalent to the statistical problem of estimating the mode of a distribution using only a finite memory for vote tallies, and is a generalization of the familiar 'secretary' ('maximum of a sequence,' 'beauty contest') problem. Loui's document explores this avenue for analysis. The experiments with HT implementation are to see how well the peak-sharpening provided by complementary HT performs with real images on complex shapes. Work on cache architectures (hierarchical schemes cascaded caches) is ongoing.

The VLSI design project produced a circuit for vote caching that can be cascaded to provide a cache of any length. Work on improving the efficiency and power of the design will continue this summer.

1.3 High Level Planning

In general, problem solvers cannot hope to create plans that are able to specify fully all the details of operation beforehand and must depend on run-time modification of the plan to insure correct functioning. The run-time planning idea becomes particularly important when different plan segments are being explored concurrently. These communicating segments may require sophisticated actions e.g. (do $PLAN_x$ until $PLAN_y$). These issues are being studied by [Russell] in the context of a cooperative planning and execution system for manipulation tasks. A recent effort [Ballard, 1984] is examining robot planning from a task frame perspective.

2. Computing with Connections

We are continuing our interest in problem-scale parallelism, both as a model of animal brains and as a paradigm for VLSI [Feldman et al., 1984]. Work at Rochester has concentrated on connectionist models and their application to vision. The framework is built around computational modules, the simplest of which are termed p-units. We have developed their properties and shown how they can be applied to a variety of problems [Feldman & Ballard, 1982]. More recently, we have established powerful techniques for adaptation and change in these networks [Feldman, 1982].

A major milestone was achieved with Sabbah's thesis on massively parallel recognition of Origami-world objects [Sabbah, 1982]. Sabbah's work extended the connectionist methodology to a problem domain with several hierarchical structural levels. The resulting program is, to our knowledge, the most noise-resistant system for dealing with this level of complexity. One outcome of Sabbah's effort has been a project to build a general purpose simulator for massively parallel systems [Small et al., 1982].

The general connectionist simulator has been well tested and is being used in a number of applications. One project involves a quite detailed simulation of motor control networks of the occulo-motor system [Addanki, 1983]. Another application is to a spreading activation model of word sense disambiguation and related problems in natural language understanding [Cottrell & Small, 1983]. A major new effort involves modelling conceptual knowledge (such as that needed for high level vision) in connectionist terms [Feldman & Shastri, 1984; Shastri & Feldman, 1984].

A new effort has been the development of a much faster C version of the simulator and the exploration of its use on a highly parallel machine. We have received major funding for a new highly parallel computer which will be available for work in IU tasks.

For a VLSI design course, a circuit was designed to implement key aspects of the "connectionist" computational paradigm [Rainero & Kautz, 1983]. This cited document is a course project report, and the exercise was mainly useful in isolating particular technical problems that must be addressed in any such parallel, activation-passing computer.

3. Motion

Our interest in motion has centered around methods for extracting rigid body parameters from optic flow and intensity images. These parameters are extremely useful in navigation and target tracking. Currently these nine parameters (origin, translational velocity, rotational velocity) can be extracted from flow via a Hough technique [Ballard & Kimball, 1983]. A more recent model exploits multiple channels [Bandyopadhyay, 1984]. We are also pursuing the use of these parameters to speed up the flow computations themselves [Stuth et al., 1983]. A major current effort relates optical flow information to surface orientation [Aloimonos & Brown, 1984] and sensor motion [Aloimonos & Brown, 1984].

4. Shape

The description and recognition of complex shapes continues to be a major focus of the project. The analysis of the dot product space representation has been improved to handle certain pathological cases, and has been generalized to accommodate different criteria for the goodness of the representation.

This simple concept of shape has been applied to the problem of reconstructing three-dimensional surfaces from

very sparse data. The key idea is to use appropriate shape descriptors to hypothesize a transformation which accounts for the difference in shape between successive contours. When the hypothesized transformation is minor, very simple-minded surface reconstruction techniques are sufficient. When there are major differences in shape or position between successive contours, our method hallucinates new contours, using the hypothesized shape transformation [Sloan & Hrechanyk, 1981]. A major new effort is the extraction and use of symmetries in images [Freidberg & Brown, this Proceedings].

Hierarchical descriptions of shapes were considered in [Ballard & Sabbah, 1981] in a preliminary fashion. Our previously reported shape model [Hrechanyk & Ballard, 1982] concentrated on problems of view-invariance and attention shifting within a single prototype. This model has been extended to handle the problems of extracting primitive shape descriptions from noisy images. Our work was motivated by dissatisfactions with smoothness criteria for intrinsic image computations. Recent work extends these ideas to simple 3-D shapes [Ballard et al., 1984].

The practicality of shape from shading computations and their interaction with the determination of other image parameters (such as illuminant position) was addressed by two papers in the Fall, 1982 DARPA Image Understanding Workshop. We are now applying the algorithm to real images, and want to investigate scenes with non-Lambertian reflectance functions that are unknown apriori. We want to explain how humans in fact use shading to derive shape, given the complexity of reflectance functions and imaging situations in the world. Two competing theories are that somehow the reflectance functions are derived fairly accurately by an adaptive procedure, or instead that we only 'support' a small number of reflectance functions that are selected by other cues (such as gloss).

5. General Theory of Vision

Work in our laboratory, among others, has demonstrated strong links between powerful IU techniques and computations used by animal visual systems. We have established strong ties with a wide range of visual scientists at Rochester and a variety of collaborative efforts are underway. One early project is to survey the computational similarities in natural and computer vision [Ballard & Coleman, 1983].

We have begun to exploit Rochester neurobiology expertise in order to hone and improve our connectionist modelling efforts. One difficult avenue is to specify the interface between our computational models and the state-of-the-art neurobiological picture. Our efforts in this direction are summarized in [Ballard & Coleman, 1983] and the collaboration is continuing. Another effort is our attempt to develop a general framework for theories of vision that would provide a common structure for integrating studies from various disciplines [Feldman, 1982].

References

- Addanki, S., "On a Distributed Approach to Oculomotor Control," TR121, Computer Science Dept., University of Rochester, 1983.
- Aloimonos, J. and C.M. Brown, "The Relationship Between Optical Flow and Surface Orientation," *Proceedings*, 7th ICPR, Montreal, August, 1984.
- Aloimonos, J. and C.M. Brown, "Direct Processing of Curvilinear Sensor Motion From a Sequence of Perspective Images," *Proceedings*, 1984 IEEE Workshop on Computer Vision Representation and Control, Annapolis, Md., May, 1984, 72-77.
- Ballard, D.H., "Cortical Connections: Structure and Function," TR133, Computer Science Department, University of Rochester, July 1984; submitted to *Behavioral and Brain Sciences*.
- Ballard, D.H., "Task Frames in Robot Manipulation," *Proceedings*, AAAI-84, August, 1984.
- Ballard, D.H., "Parameter Networks: Towards a Theory of Low-Level Vision," *Proceedings*, 7th IJCAI, Vancouver, British Columbia, August 1981.
- Ballard, D.H. and P.D. Coleman, "Cortical Connections: Structure and Function," presented at U. Mass. Workshop on Vision, Brain and Cooperative Computation, Amherst, Mass., May 1983.
- Ballard, D.H. and O.A. Kimball, "Rigid Body Motion from Depth and Optical Flow," *CVGIP Special Issue on Computer Vision*, 1983; also TR70, Computer Science Department, University of Rochester, November 1981.
- Ballard, D.H. and D. Sabbah, "Detecting Object Orientation from Surface Normals," *Proceedings*, 7th IJCAI, Vancouver, British Columbia, August 1981.
- Ballard, D.H., A. Bandyopadhyay, J. Sullins and H. Tanaka, "A Connectionist Polyhedral Model of Extrapersonal Space," *Proceedings*, 1984 IEEE Conference on Computer Vision, Annapolis, MD., May, 1984.
- Bandyopadhyay, A., "A Multiple Channel Model for Perception of Optical Flow," *Proceedings*, 1984 Workshop on Computer Vision Representation and Control, Annapolis, Md., May, 1984, 78-82.
- Bandyopadhyay, A., "Interest Points, Disparities and Correspondence," *Proceedings*, DARPA Image Understanding Workshop, New Orleans, La., 1984.
- Brown, C.M., "Mode Estimation with Small Samples and Unordered Bias," TR138, Computer Science Department, University of Rochester, June 1984.
- Brown, C.M., "Hierarchical Cache Accumulators for Sequential Mode Estimation," TR125, Computer Science Department, University of Rochester, July 1983.
- Brown, C.M., "Bias and Noise in Hough Transform: Theory," *Pattern Analysis and Machine Intelligence*, 1983; also, TR105, Computer Science Department, University of Rochester, July 1982.
- Brown, C.M., M. Curiss, and D. Sher, "Bias & Noise in Hough Transform: Experiments," *Proceedings*, IJCAI-83, Karlsruhe, West Germany, August, 1983; also TR113, Computer Science Dept., University of Rochester, 1982.
- Brown, C.M. and D. Sher, "Modeling the Sequential Behavior of Hough Transform Schemes," *Proceedings*, DARPA Image Understanding Workshop, November, 1982; also TR114, Computer Science Department, University of Rochester, August 1982.
- Brown, C.M. and J.A. Feldman, "Statistical Questions Arising in the Use of Hough Techniques in Image Understanding," *Proceedings*, ONR Workshop on Statistical Image Processing and Graphics Workshop, Luray, VA., 24-27 May 1983.
- Cottrell, G.W. and S.L. Small, "A Connectionist Scheme for Modelling Word Sense Disambiguation," *Cognition and Brain Theory*, 6 (1), 89-120, 1983.
- Feldman, J.A., "A Connectionist Model of Visual Memory," in *Parallel Models of Associative Memory*, G.E. Hinton and J.A. Anderson (eds.), Hillsdale, NJ: Lawrence Erlbaum Associates, publishers, 1981.
- Feldman, J.A., "Dynamic Connections in Neural Networks," *Biological Cybernetics*, 46, 27-39, 1982.
- Feldman, J.A., "Four Frames Suffice: A Provisionary Model of Vision and Space," presented at the U. Mass. Workshop on Vision, Brain and Cooperative Computation, Amherst, Mass., May 1983; also, TR99, Computer Science Dept., University of Rochester, September 1982.
- Feldman, J.A. and D.H. Ballard, "Connectionist Models and Their Properties," *Cognitive Science*, 6, 205-254, 1982.
- Feldman, J.A. and L. Shastri, "Evidential Inference in Activation Networks," *Proceedings*, Cognitive Science Conference, Boulder, Co., July 1984.
- Feldman, J.A., D.H. Ballard, C.M. Brown and S.L. Small, "Rochester Connectionist Papers, 1979-1984," TR124, Computer Science Department, University of Rochester, June 1984.
- Friedberg, S.A., "Finding Axes of Skewed Symmetry," TR127, Computer Science Department, University of Rochester, February 1984.
- Friedberg, S.A. and C.M. Brown, "Symmetry Evaluators," *Proceedings*, DARPA Image Understanding Workshop, New Orleans, LA, 1984.
- Friedberg, S.A. and C.M. Brown, "Finding Axes of Skewed Symmetry," *Proceedings*, 7th ICPR, Montreal, August, 1984.

- Hrechanyk, L.M. and D.H. Ballard, "Viewframes: A Connectionist Model of Form Perception," *Proceedings, DARPA Workshop*, June 1983.
- Hrechanyk, L.M. and D.H. Ballard, "A Connectionist Model of Form Perception," *Proceedings, Workshop on Computer Vision: Representation and Control*, Rindge, New Hampshire, August 1982; also, *Computer Science and Engineering Research Review*, Computer Science Department, Fall, 1982.
- Lampeter, W., "Design, Function and Performance of a System that Screens Chest Radiographs for Tumors," *Proceedings, 1st CVPR Conference*, June 1983.
- Lampeter, W., "Three Image Experts Which Help Distinguish Lung Tumors From Non-Tumors," TR123, Computer Science Department, University of Rochester, February, 1984.
- Loui, R., "How Fast Hough?" Internal course project report, Computer Science Dept., University of Rochester, April 1983.
- Rainero, E. and H. Kautz, "A Connection Chip," Internal course project report, Computer Science Dept., University of Rochester, April, 1983.
- Russell, D.M., "Schema-based Problem Solving," forthcoming Ph.D. Dissertation, Computer Science Dept., University of Rochester, December 1984.
- Sabbah, D., "A Connectionist Approach to Visual Recognition," TR107, Computer Science Dept., University of Rochester, April 1982; also Ph.D. thesis, Computer Science Dept., University of Rochester, April 1982.
- Shastri, L. and J.A. Feldman, "Semantic Networks and Neural Nets," TR131, Computer Science Department, University of Rochester, June, 1984.
- Sher, D. and A. Tevanian, "A Hough Chip," Internal course project report, Computer Science Dept., University of Rochester, April 1983.
- Small, S.L., L. Shastri, M.L. Brucks, S.G. Kaufman, G.W. Cottrell and S. Addanki, "ISCON: A Network Construction Aid and Simulator for Connectionist Models," TR109 Computer Science Dept., University of Rochester, September 1982.
- Sloan, K.R., Jr. and L.M. Hrechanyk, "Surface Reconstruction from Sparse Data," *Proceedings: Pattern Recognition and Image Processing*, Dallas, Texas, August, 1981.
- Stuth, B.H., D.H. Ballard and C.M. Brown, "Boundary Conditions in Multiple Intrinsic Images," *Proceedings, IJCAI-83*, Karlsruhe, West Germany, 1983.

IMAGE UNDERSTANDING RESEARCH AT USC: 1983-84¹

R. Nevatia
Intelligent Systems Group
Departments of Electrical Engineering
and Computer Science
University of Southern California
Los Angeles, California 90089-0272

1. ABSTRACT

Our research is divided into three major areas: 3-D vision, mapping from aerial images and parallel processing. In 3-D vision, we are exploring the use of curvature properties for describing surfaces, and techniques of computing generalized cones from sparse range data. The work on mapping concentrates on stereo analysis and an "expert" module for extracting runways in a major airport complex. The parallel processing work has consisted of the design of a new cellular architecture and mapping of selected algorithms on it.

2. INTRODUCTION

Our research in the recent period is divided into three major areas:

- a) 3-D "Robotics" Vision
- b) Mapping from Aerial Images
- c) Parallel Processing of IU Algorithms

Our activities in each of the areas is summarized below. Other papers in these proceedings from our group provide more details of some of the specific research, in which case only a brief note is provided below. This paper describes the work of many individuals in our group. The names of the people contributing to each specific section are given in the text.

3. 3-D VISION

The aim of this research is the description of 3-D objects and surfaces, for the purposes of recognition, position and orientation determination, and inspection. 3-D representations can describe either the surface or the volume of an object. Volume descriptions have strong advantages over surface descriptions in most cases. However, some smooth surfaces, e.g. a turbine blade, are "featureless" and possess little volume and may be better described as surfaces. We are pursuing both kinds of descriptions.

3.1. 3-D Surface Representation (Medioni and Nevatia)

A common method for representing surfaces is by piecewise approximation with some bases surfaces. Such methods do not give "natural" descriptions and the places where the piecewise descriptions meet may have no clear physical properties. We suggest that even featureless surfaces have some distinguishing points or lines on them, and that these "features" are appropriate for matching and for generating higher level descriptions. We propose that high curvature points and curves joining such contiguous points are one class of such features. Such points are on the occluding contours, surface discontinuities and on the "ridges" and "valleys" of the surfaces.

This approach and some results are described in a separate paper in these proceedings [1]

3.2. Generalized Cone Descriptions (Nevatia, Kashipati and Medioni)

Generalized cones have come to be increasingly recognized as a powerful representation scheme for 3-D volumes. However, the number of implementations that compute them from scene data is rather small, and they have severe limitations. Typically, they assume perfect 3-D data and/or knowledge of specific objects in the scene. We have initiated work at computing such descriptions from sparse 3-D data, as might be generated by an edge-line based stereo system (prior to interpolation). Currently, our method applies only to a restricted class of generalized cones and is not fully tested, but we expect it to generalize to a much broader class of objects.

The boundaries (or edges) in any scene can be considered to be in the following classes, following the traditional classification introduced by Huffman and Clowes [Ch. 4 refs. 4&5] (see Figure 1 for an example):

1. Occluding boundaries: This is a boundary where the visible surface is all to one side of the boundary (shown by "1" in Figure 1). Previous contour analysis, of Nevatia and Binford, and of Marr, essentially assumes that these are the only visible contours.
2. Surface slope discontinuities: These may be caused by slope discontinuities in the cross-section shape, such as along the corners of the cross-sections of a polyhedral object or by a "terminator" (i.e the end) of a generalized cone (shown by "2" in Figure 1). These kinds of boundaries would cause difficulties in the earlier methods of analysis, but we will show how they may provide very valuable pieces of infor-

¹ This research was supported by Defense Advanced Research Projects Agency under contract numbers F33615-82-K-1786 and F33615-84-K-1404, monitored by Wright Patterson Aeronautical Labs

mation.

3. Surface Markings- These are caused by changes in the surface reflectance rather than the surface position or slope ("3" in Figure 1). In simplistic analysis, these may be confused with occluding boundaries.
4. Others- Other sources are due to noise, shadows, highlights etc. Our approach does not deal with them explicitly, but should work in their presence (we can essentially consider them to be same as surface markings for our analysis).

For generalized cones, the important boundaries could be alternatively classified as those produced from "contour generators" and terminators. Intuitively, contour generators are the extremal points on the surface which enclose the visible surface (and are thus view-point dependent). For a smooth generalized cone, the contour generators are the points on the surface where the line of sight is tangential to the viewed surface. More generally, the contour generators are the loci of the extremal points on the cross-sections. Note that contour generators are a subset of the occluding boundaries. (In this, our definition is slightly different from that used in Shafer [2], but has the same intuitive notion. Also we will not differentiate between "contours" which are projections in the image of contour generators and contour generators, here as we use 3-D boundaries.)

The terminators of a generalized cone are simply its ends (imagine an infinite generalized cone that has been cut at two ends). Note that the cut, and hence a terminator need not be planar, and when planar, it need not be normal to the axis. Thus, we prefer to describe a right, circular cone with a slanted cut, as a straight, homogeneous generalized cone with an oblique termination, rather than as an homogeneous, generalized cone with cross-section changing shape at the end, i.e. our descriptions are necessarily in terms of right generalized cones. Note that the terminator may share part of the occluding boundary. The terminators have been a source of difficulty in analysis of boundaries, as in [3]; however, they can also provide valuable clues to the shapes of the cross-sections.

Now, the scene description problem may be considered to be that of isolating the contour generators and terminators of the generalized cones present in a scene. This axis of the generalized cone is the axis of symmetry of the contour generators, and the terminators give cross-section shape under certain conditions.

The key to our approach consists of the following observations about boundaries of generalized cones:

1. A contour generator is tangential (in 3-D) to the terminator boundaries. (3-D tangency also implies a 2-D tangency). Further, the contour generator must be to one side of the plane containing the local contour tangent and the viewing point. (In 2-D, The terminator boundary must be all on one side of the contour at the junction.)
2. For a linear, straight, homogeneous generalized cone (LSHGC), the contour generator is planar from any view (established by Shafer [2]). For a non-linear SHGC, the contour generators are planar in a side

view, but not necessarily in an oblique view. If we view a non-straight, non-homogeneous generalized cone as consisting of piecewise LSHGCs, then we can expect the contour generators to be "piecewise coplanar". In our current implementation, we have tested only the LSHGCs, but believe that we can extend to elongated generalized cones by using the piecewise approximations.

The method, then, consists of computing descriptions from a given set of segments that follows the above constraints. In our current implementation, we find all possible pairs of contour generators (i.e. all coplanar pairs), then compute the corresponding terminators and then evaluate the descriptions. Clearly, for a complex scene, it will not be feasible to examine all alternatives, and some choices will need to be based on partial analysis alone. Tracing of terminator boundaries is relatively straight-forward when there are no gaps, otherwise only fragments of terminators will be obtained.

The selection of descriptions is based on the following:

1. Longer contour generators are preferred over shorter ones (i.e. we prefer descriptions with longer axes).
2. Parallel contour generators are preferred (i.e. cylinders are preferred over cones).
3. Descriptions with closed and/or planar terminators are preferred.

With this simple implementation, we are able to find good descriptions for simple objects such as cylinders and cones. We have analyzed scenes with some occlusion and surface markings, but with no missing segments.

4. MAPPING FROM AERIAL IMAGES

This is part of our continuing research from previous years. In the past, we have developed methods for linear feature extraction and region segmentation, image to map correspondence, stereo analysis, and shadow analysis. These have been reported in previous DARPA Image Understanding workshops, our internal reports and other open literature. Our current work focusses on two projects: stereo analysis and development of an "expert" module for airport analysis.

4.1. Stereo Analysis (Mohan, Medioni and Nevatia)

Stereo is relied upon heavily by human analysts in mapping from aerial images. In many cases, stereo may be the only direct cue for obtaining 3-D data in aerial images (direct ranging is often not practical). In previous work, we reported a stereo algorithm that matches the line segments detected in the two images [4]. We feel that being an edge based scheme, it has important advantages over the conventional intensity correlation based stereos, and also over methods that match unconnected edges. Our recent work has been in a detailed testing and evaluation of this algorithm on a variety of scenes. One of the complex images we have been working with is the "Pentagon" image shown in Figure 2 (resolution 512 x 512); this is a good test image due to the presence of parallel

lines, and fine detail, and there is some comparative data available from other work [5]. Our tests have indicated several minor and major areas for further improvement:

1. We found that imposing a constraint that the contrast of the matching segments be similar caused our system to loose many good matches. We have eliminated this constraint but still use the sign of the contrast. This also indicates that correlation stereo would have difficulties in these instances too.
2. Order preservation - our original algorithm did not explicitly require any order to be preserved among the matching segments. We have incorporated an order constraint and tested its effects. Our algorithm works with segments rather than edges, and no complete order can be defined on segments in 2-D. We define and use a pairwise order instead - the order of two segments is defined by their relative positions in the parts where they overlap in the direction of the epipolar lines. This order is well defined and unambiguous as segments may not intersect. The order is used in matching in the following way: evaluation of a matching pair (i,j) depends on how well the disparity of (i,j) agrees with the disparities of other matching segments pair (k,l) in the neighborhood of i and j ; if the order of (k,l) is not the same as that of (i,j) then (k,l) contributes a large penalty weight to the evaluation of (i,j) . (We are omitting the details here as the evaluation function is rather complex; it is described in detail in [4]). Our experience with using the order constraint indicates that the use of order removes some incorrect matches, and finds matches for some segments left unmatched previously. However, the improvements seem to be rather minor, at least for the examples we have tested on. Perhaps, this is an indication of the robustness of the original, unordered algorithm itself.
3. The third area of improvement has been in the implementation of a hierarchical matching method. As the resolution of the image increases, so does the number of segments detected in it. In our case, since we use 2-D windows to establish context, the number of segments in a window grows rapidly with resolution. We can control this complexity by using the information from a lower resolution match to guide a higher resolution match.

A match of a certain resolution gives disparities of the points on the segments that match. These disparities are used to compute disparities for in-between points by linear interpolation. These interpolated values provide the initial disparity for the match at the next higher level. Since the approximate disparity is known, the search for a match is now restricted to a small range, thus limiting the computational complexity and also reducing chances of an error. Note that the errors made at an earlier stage will not be corrected at higher resolution by this method, though they do not necessarily propagate (the higher resolution segments may end up with no matches).

Figure 3 shows the segments extracted and matched from Figure 2, reduced to a resolution of 128×128 . The number of matched segments is 359. Figure 4 shows the segments matched at the resolution of 256×256 , the number of matched segments is now 1340. We believe that the matching results for this image are impressive and almost all segments are matched correctly. However, we still lack a suitable way of displaying the results in 3-D as linear interpolation is sensitive to local errors which mark the other data in a perspective display. As a partial solution, Figure 5 displays the points with disparity values that would put their height a certain minimum amount above ground (about half the height of the building itself). It is clear that major parts of the pentagon building are detected correctly.

Some Remaining Problems

- Our most important problem is the detection of segments that are matched incorrectly. The number of such mismatches is small, but some of them can have a significant effect on the interpretation of the image. We are currently investigating use of more global context to identify and correct such matches.
- We need to develop a suitable way of interpolating and displaying surfaces. Many sophisticated methods of surface interpolation exist in the literature (e.g. Terzopoulos [6]), our main difficulty is in identifying errors and discontinuities.

4.2. Developing "Domain Experts" (Huertas, Nevatia and Price)

As a step towards developing visual domain experts, we have selected the task of mapping a major commercial airport complex. Our first chosen task is to find the runways and taxiways. One may think that these objects would be easy to extract and would consist of relatively homogeneous, elongated linear strips. In reality, these objects are rather complex, an example of the Los Angeles International Airport (LAX) is shown in Figure 6. The intensity of a runway surface is not uniform, in fact, in one of the runways the surface material itself changes from concrete to blacktop, perhaps due to extensions made at different periods. There are many markings on the runways and taxiways - some are intended to aid pilots, others are due to factors such as tire tread marks, dirt, etc. The runways and taxiways have many intersections and the contrast with the shoulders is not always strong.

The difficulty of mapping in face of the above complexities is, of course, dependent on the amount of a priori knowledge that is assumed. If we assume that we are looking at LAX and have a previous map, the task of locating runways is much easier. We are taking a mid-way approach that assumes that we are looking at a major commercial airport and know the altitude of the camera, but not the identity of the scene. Thus, we can infer a range for widths and lengths of the runways and taxiways but do not know their specific relative locations and orientations.

Our approach is to first extract linear line segments in the image using our "LINEAR" software (which incor-

porates both the Nevatia-Babu [7] and the Marr-Hildreth [8] edge detectors). Figure 7 shows the segments extracted from Figure 6 (the original image is 3000 x 800.) Then we group the line segments into "anti-parallel" pairs or APARS (parallel lines of opposite contrast or orientation). If the runways followed the simple model of being homogeneous, linear strips, we could expect the APAR grouping (assuming a known range of widths) to isolate them. With the complications described above, we can expect to get APARs not only between the two edges of the runways, but between the edges, the markings, the treadmarks etc, and also between these features and surrounding features such as edges of the shoulder area (the full set of APARs is not shown, as the picture is too cluttered to convey information easily).

Now, our task is to separate the desired APARs from the host of the others. Also, there is no single APAR that spans the entire length of the runway and we must join the appropriate fragments. At this stage, we need to use our knowledge of the airports. Our implementation is in very preliminary stages; we will describe our approach first and then the specific implementation.

Broadly speaking, we wish to follow a "generate and test" paradigm. Then, the work to be done is in rules and methods for hypothesizing and verifying for this particular problem. One way of hypothesizing is to search for an area of least ambiguity and to propagate the inferences. For example, if in a certain part of the runway (usually at one end that gets little traffic) we find only a single APAR that has the appropriate width, we can try to extend it. Another method is to combine evidences from various pieces and prefer those that have broad support. For example, if a set of APARs are colinear, we can hypothesize a longer APAR and try to verify it. Verification might be by checking if we can explain the departure of the data from the ideal by the expected causes. So far, we have implemented only simple hypothesis and verification methods which are described below.

The first step is to find those APARs that may be part of a runway or taxiway. The direction of the runways may be known a priori, if we are analyzing a known airport. In our case, we constructed a length weighted histogram of segment directions and used the peak to get an approximate direction for the runway (we are assuming that the scene around runways is dominated by lines along and on the runway). We also estimate minimum and maximum widths for runways and taxiways. Again, these should be easily available for a known airport, or even for a known type of airport. We derived these widths from a length weighted histogram of APAR widths. The set of potential APARs that might be part of runways is then constructed by choosing those that are in the appropriate direction and have widths between the required bounds. Further, they are required to have an aspect ratio exceeding 5 to 1. Figure 8 shows such a set constructed from Figure 7 (the display shows the APAR as a rectangular box).

The next stage consists of several steps. Those APARs that share a common segment, and are colinear and of same "color" (brighter or darker than surround) are hypothesized to form a new longer APAR (the assumption

is that we are likely to miss one side of a runway due to intersections or other reasons). A test on the acceptability of the merged APAR is also performed (described below) and APARs that are completely contained in another are eliminated. Figure 9 shows the results of the processing after this stage.

The runways are still in many disconnected APARs. The next stage is to hypothesize that colinear APARs are connected. Before a connection is established, we examine the gap between two APARs to be connected, looking for evidence contrary to their being connected. Figures 10 (a) and (b) show a connection hypothesis (the area to be bridged is shown in bold dashes) and the segments in this area. At this time, the verification consists simply of examining whether there are many segments normal to the direction of APAR, as they would represent a potential obstacle (the exact measure used is a ratio of the sum of the lengths of the segments in the near normal directions to the sum of the lengths of the segments in the APAR direction). The verification process needs to be made much more informed and not dependent on a single number as this. Figure 11 shows the four longest resulting APARs and include the desired runways (the top and the bottom). We have also obtained similar results for extracting taxiways (not shown).

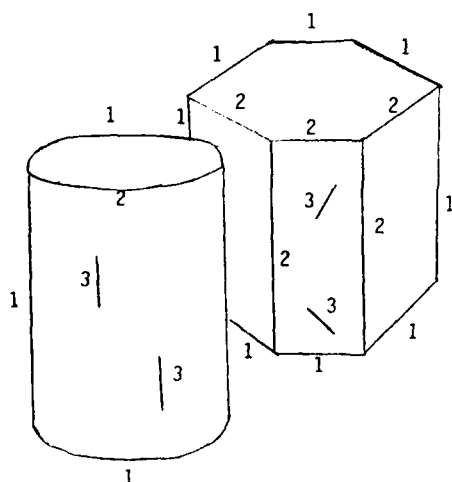
This example is not intended to indicate that we have developed a general system that is capable of extracting runways and taxiways in complex images, but simply to show the complexity of the problem and the value of certain types of processing.

5. PARALLEL PROCESSING OF IU ALGORITHMS (Moldovan, Dixit, and Tenorio)

In the past year, we have initiated a serious study of the parallel processing of IU algorithms. Our approach has been to take specific algorithms that are representative of broad class of IU algorithms and study their parallel implementation. We have been concentrating on the "higher level" algorithms. Our studies have been mostly in the context of a parallel architecture, called SNAP, proposed at USC by Prof. Moldovan and his students. SNAP consists of a 2-D array of processors with each processor connected to its four neighbors. There is also a global bus for broadcasting to all processors simultaneously. We have looked at the mapping of discrete relaxation and general production systems on this machine. The results are described in another paper in these proceedings [9]. The design of SNAP is not complete at this time, and we can only get a preliminary estimate of the suitability of this type of architecture for certain types of IU algorithms. Complete complexity analysis of a complex vision algorithm on a complex, parallel machine remains a difficult task; our efforts represent very early attempts.

References

1. Medioni, G. and Nevatia, R., "Description of 3-D Surfaces Using Curvature Properties," *Proceedings of Image Understanding Workshop*, October 1984.
2. Shafer, S.A., "Shadow Geometry and Occluding Contours of Generalized Cylinders," Tech. report, CMU Report CS-83-131, May 1983.
3. Nevatia, R. and Binford, T.O., "Description and Recognition of Complex-Curved Objects," *Artificial Intelligence*, Vol. 8, 1977, pp 77-98.
4. Medioni, G. and Nevatia, R., "Segment-based Stereo Matching," *Proceedings of DARPA Image Understanding Workshop*, Washington, D.C., June 1983.
5. Ohta, Y. and Kanade, T., "Stereo by Intra and Inter-scanline Search Using Dynamic Programming," Tech. report, CMU-CS-83-162, October 1983.
6. Terzopoulos, D., *Multiresolution Computation of Visible-Surface Representations*, PhD dissertation, Massachusetts Institute Technology, Departments of Computer Science and Electrical Engineering, 1984.
7. Nevatia, R. and Babu, K.R., "Linear Feature Extraction and Description," *Computer Graphics and Image Processing*, Vol. 13, 1980, pp. 257-269.
8. Marr D. and Hildreth E., "Theory of Edge Detection," *Proceedings of the Royal Society of London*, B207, 1980, pp 187-217.
9. Dixit, V. and Moldovan, D.I., "Semantic Network Array Processor and Its Application to Image Understanding," *Proceedings of Image Understanding Workshop*, 1984.



- 1 - occluding boundaries
- 2 - surface discontinuities
- 3 - surface markings

Figure 1. Two occluding objects and a classification of the boundaries



Figure 2: A stereo pair of images
(courtesy of Dr. Kanade)
(a) left (b) right

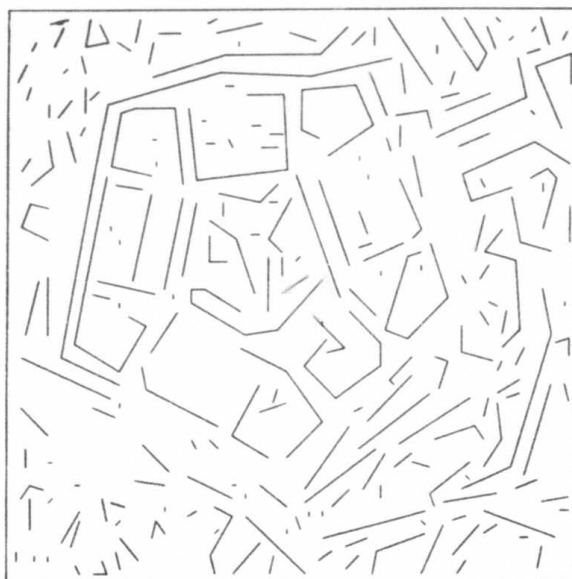


Figure 3: Segments detected and matched in Figure 1 at
128x128 resolution
(a) left (b) right

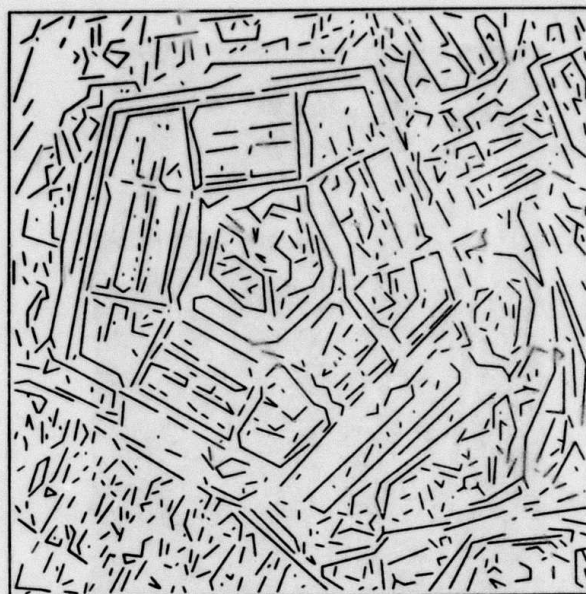


Figure 4: Segments detected and matched if Figure 1 at
256x256 resolution
(a) left (b) right

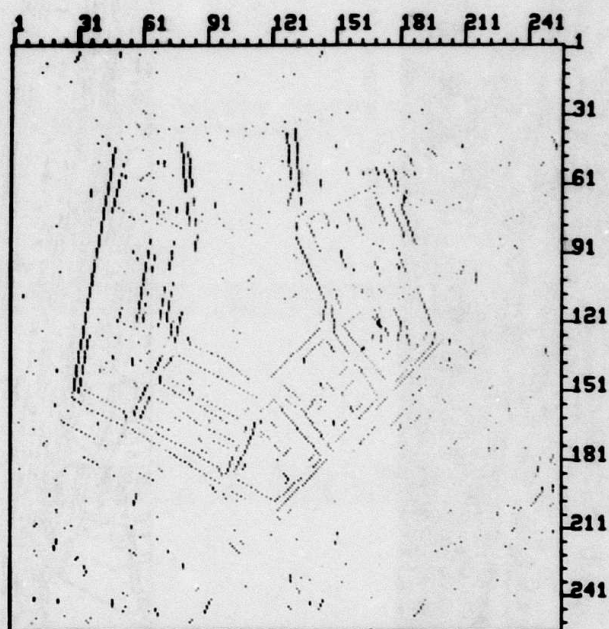


Figure 5: Points that are at least a certain height above
ground

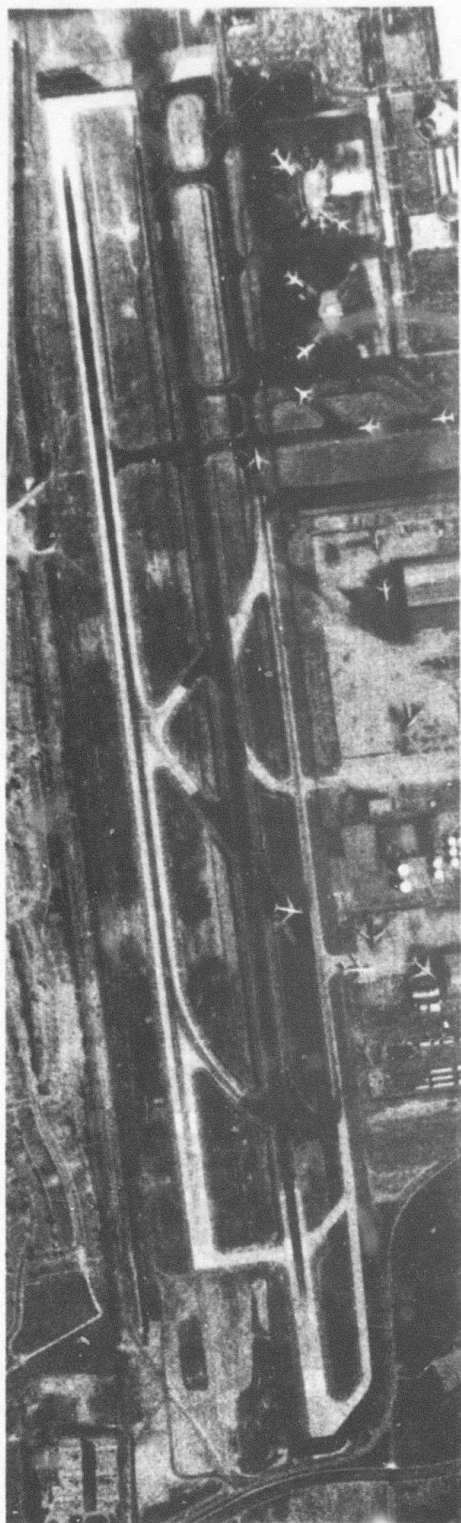


Figure 6: Image of LAX Airport



Figure 7: Segments detected from Figure 5

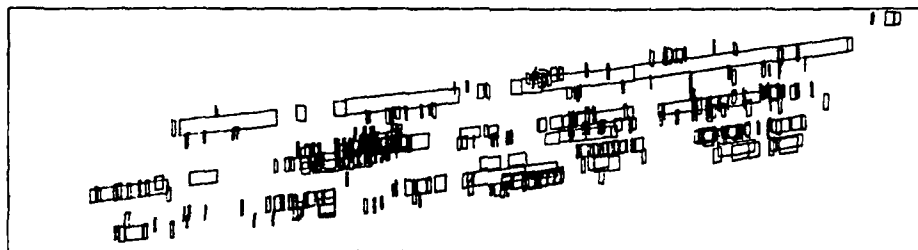


Figure 8: Potential runway APARS (displayed as "boxes")

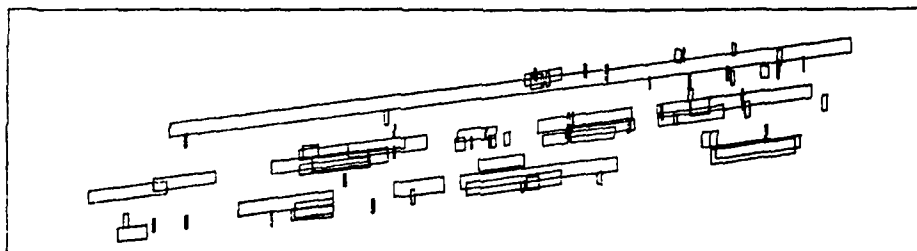
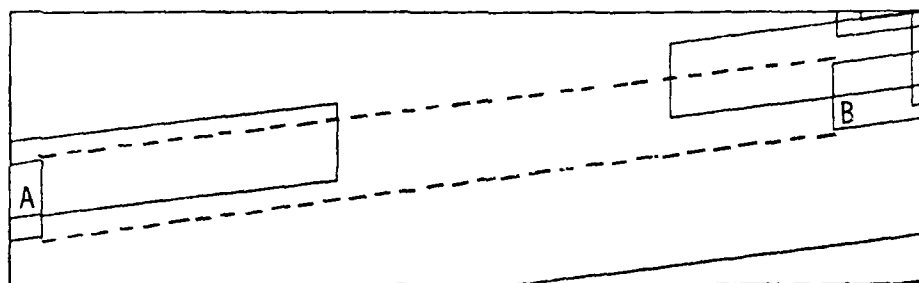
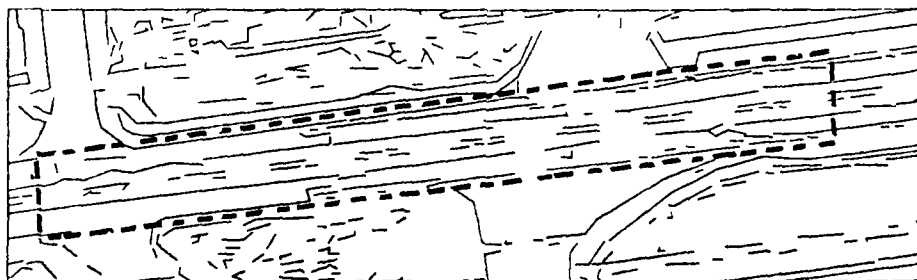


Figure 9: Connected runway APARS



(a)



(b)

Figure 10 (a): A connection hypothesis and area to be bridged (shown by dashed box)
(b): Segments in and around the area to be bridged

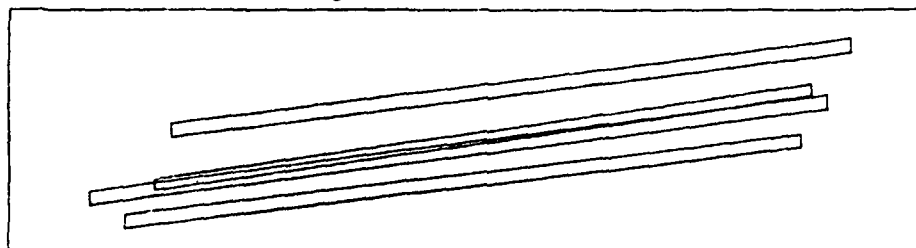


Figure 11: Four longest APARS after the final processing stage

Image Understanding Research at CMU

Takeo Kanade

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

In the CMU Image Understanding Program we have been working on both the basic issues in understanding vision processes which deal with images and shapes, and the system issues in developing demonstrable vision systems. This report reviews our progress since the June 1983 workshop proceedings. The highlights in our Program include:

- *Shafer has been working on techniques for dealing with shadows, reflectance, color and highlights.*
- *Ohta and Kanade have developed a stereo program using dynamic programming techniques.*
- *Thorpe and Matthiess have demonstrated a vision-based navigation system with obstacle avoidance on the CMU Rover.*
- *Lucas has applied the method of differences in obtaining the camera motion from an image sequence. Kanade has been developing a new theory for motion from image sequences.*
- *Kanade, Herman, Tomita, and Smith have been developing techniques for generating and matching scene descriptions from 3D range imagery and aerial photos.*
- *Webb and Dew have been working toward a vision system on a CMU-designed systolic machine called Warp.*
- *McKeown has continued to work in the digital mapping and photo interpretation area.*

1. Shapes and Images

1.1. Shadow Geometry for Solids of Revolution

We have been producing an implementation of Shafer's theories [19] of shadow geometry and occluding contour interpretation for solids of revolution. We now have a collection of programs that produce and analyze occluding boundaries and shadows of solids of revolution. Preliminary results indicate that, for an image of a solid of revolution with shadow quantized to a 100 x 100 grid, the surface normals on a solid of revolution can be estimated to an accuracy of 1% in the p and q coordinates in

gradient space, using the methods of Shafer's thesis. We also have a negative result: apparently, there is not sufficient information in the surface normals along the terminator to uniquely compute the angle of view (angle between the line of sight and the axis) in an image of a solid of revolution. However, alternative methods can be used for finding this angle, such as examining the elliptical images of the ends of the object. We have assembled a set of images of real solids of revolution, and are implementing a hand-segmentation program to produce contours from these images that can be used as input to the existing analysis programs.

1.2. Reflectance Maps Under Perspective

We have been examining the use of reflectance maps in images taken under perspective projection. The effect of the $[\cos^4 \theta]$ term, previously noted by Horn and Sjöberg [6], seems to be smaller than the effect of the varying viewing direction upon the photometric angles e and g . For non-Lambertian surfaces, the specular peak in the reflectance map thus varies considerably in size, shape, and position in gradient space. This suggests that, for precise photometric image interpretation, the reflectance map must be determined for each pixel individually. We are exploring the magnitude of this effect with synthetic data, including the possibility of algorithms based on compromises such as "lazy evaluation" of the reflectance map or the use of a collection of reflectance maps, each for some rectangular region of the image and representing the "average" reflectance map for that region.

1.3. Color and Highlights

Color and highlights are important image features, yet very little research has been done on them from the computer vision point of view. We have produced a theory [20] that describes the relationship between color and highlights in an image of a glossy surface. Because specular and diffuse reflection are caused by different physical processes (interface versus deep reflection),

they produce different colors in the image. The pixel values measured by the camera will be a sum of these two components, i.e. a linear combination of the R-G-B color of the specular reflection and the color of the diffuse reflection at each point. Since these colors are constant across a surface, the pixel values measured across a single surface will form a parallelogram in R-G-B color space. The position of any pixel's color within that parallelogram yields the magnitude of the specular and diffuse reflection at that point. This theory applies to rough surfaces of inhomogeneous media, i.e. most plastics, paints, glazed ceramics, glass, etc. The color theory will work under extended light sources, perspective imaging projection, and curved surfaces. We are working now to extend the theory to deal with diffuse ("ambient") illumination, and we hope to calibrate a camera to use in implementing the theory on real images.

1.4. Stereo

We have implemented a stereo algorithm using dynamic programming [18]. Edge-based stereo involves two searches. One is *intra-scanline* search for obtaining correspondence of edge-delimited intervals on each scanline pair after the pair of images has been rectified so that epipolar lines are horizontal. The intra-scanline search can be treated as the problem of finding a matching path on a two dimensional (2D) search plane. The other is *inter-scanline* search for possible correspondence of vertically connected edges across scanlines in right and left images. This provides the consistency constraints that inter-scanline matchings should satisfy. Previous use of dynamic programming has been limited to the intra-scanline search [4, 1]. Henderson processed pairs of scanlines sequentially where results of the previous line guided searches in the succeeding scanline. Baker [1] employed post-processing on the results of intra-scanline search to enforce the inter-scanline constraints.

We utilize dynamic programming for both of the inter-scanline and intra-scanline searches, and the two searches proceed simultaneously: the first supplies the consistency constraint to the second while the second supplies the matching score to the first. An interval-based similarity metric is used to compute the score. By simultaneously considering intra- and inter-scanline searches, the correspondence problem in stereo can be cast as that of finding in a three dimensional search space an optimal matching surface that most satisfies the intra-scanline matches and inter-scanline consistency. Here, a matching surface is defined by stacking 2D matching paths, where the 2D matching paths are

found in a 2D search plane whose axes are left-image column position and right-image column position, and the stacking is done in the direction of the row (scanline) number of the images. The cost of the matching surface is defined as the sum of the costs of the intra-scanline matches on the 2D search planes, while vertically connected edges provide the consistency constraint across the 2D search planes and thus penalize those intra-scanline matches which are not consistent across the scanlines.

The algorithm has been tested with various images including urban aerial images, synthesized images, and block scenes. The results show that the 3D search achieves roughly 1/10 the error rate of the 2D search. For some images containing a large number of edges, the 3D search requires as much as 10 to 15 times longer processing. However, the processing time is expected to be reduced drastically by implementing the algorithm on a parallel machine such as a systolic array processor.

2. Vision for Navigation

The IUS group has been working on navigational vision in collaboration with the CMU-Robotics Mobile Robot Lab. The effort includes path planning, motion determination, and obstacle detection using video and sonar data [22, 23].

2.1. A Visual Navigation System

Using a CMU-rover testbed vehicle, a visual navigation system has been demonstrated by Thorpe and Matthies [10] to maneuver to a pre-defined location in a static environment. The visual system is based on algorithms developed by Moravec for the Stanford Cart. At each cart position, these algorithms used stereo correspondence in nine camera images to triangulate the distance to potential obstacles. Motion of the vehicle was determined by tracking these obstacles over time. Detailed evaluations have been made on the performance of the navigational vision system during the evolution from the Stanford Cart to the present CMU system. The results of the evaluation have led to the use of fewer images per step, to the use of more constraints to limit the search in the correspondence process, and toward the use of a different motion solving algorithm that better exploits the rigid motion of the scene.

2.2. Obtaining Camera Motions from Image Sequences

Lucas and Karade [9] applied the method of differences to the problem of obtaining camera motions from an image sequence.

The *method of differences* refers to a technique for image matching that uses the intensity gradient of images to iteratively improve the match between the two images. When the iterative scheme is combined with image smoothing, the method exhibits good accuracy and wide convergence range. This method will be used in the actual demonstration system of visual navigation.

Kanade [8] is currently working on a new theory for motion from image sequences that relates line correspondences, differential motions of the viewer, and temporal and spatial differentials of images. The theory consists of two parts. The first part provides the formulas to solve the differential motion of the viewer from line correspondences. The second part relates the line correspondence problem with temporal and spatial image differentials. Interestingly, this theory seems to provide a way to cleverly get around the aperture problem that arises as an ambiguity in matching points using only the local image properties within a window. When viewed as a line correspondence problem, local image properties do provide enough information.

3. Vision on a Systolic Machine Warp

We have started developing a vision system on a systolic array machine. The machine called *Warp* is being designed and built by H. T. Kung's VLSI group at CMU. The original design of Warp was for a parallel architecture providing high-speed floating-point computation for signal processing. This design has been changed in cooperation with the IUS group to provide the computational flexibility needed for doing a variety of low-level vision tasks.

Warp consists of ten cells organized in a linear array. Each cell contains a 32-bit floating-point adder and multiplier, 4K of 152-bit word micro-store, and 4K of 32-bit RAM, as well as other registers to provide sufficient control. The adder and multiplier, which are pipelined, each produce one floating-point result every 200 nanoseconds for a throughput of 10 MFLOPS per cell or 100 MFLOPS for the whole array. The cells include facilities for systolic processing or limited local address generation, so that each cell can be programmed individually to do a different computation, or the whole array can do a coordinated systolic computation. The 10-cell machine will be interfaced with a VAX 11/780 through an APTEC DPS-2400 interface machine, which provides 1 Mbyte of memory and 24 Mbyte/sec bandwidth.

At present, Webb is working to make Warp a useful tool for low-level vision [8]. This will involve programming a large number of low level vision subroutines, and integrating Warp into a full-scale visual task. The low-level vision subroutine library will provide a ready source of programs for someone who wants to use Warp, as well as providing numerous guides in programming Warp for a new programmer. A library of vision subroutines has been studied and classified from the point of implementing them on Warp. The assembler and simulator for Warp have been written. Several sample Warp programs including 3x3 convolutions have been written.

Dew [2] has investigated re-implementation of FIDO algorithms (visual obstacle avoidance algorithms by Moravec and Thorpe) on Warp; we anticipate a speed-up of a factor of 10 from Warp implementation compared with VAX implementation. We also plan to investigate use of Warp for other navigation tasks, including path following.

4. 3D Vision System: Generation and Matching of 3D Scene Descriptions

4.1. From 3D Range Imagery

We have been active in developing techniques for generating three-dimensional scene descriptions from range imagery and for matching scene descriptions. These techniques will be applied to both industrial vision problems and outdoor scene analysis to be used for outdoor navigation.

Two pieces of work have been done with different approaches and emphasis. The first piece of work, by Smith [21], produces object-centered three-dimensional descriptions starting from point-wise 3D range data obtained by a light-stripe rangefinder. A careful geometrical analysis shows that contours which appear in light-stripe range images can be classified into eight types, each with different characteristics in occluding vs. occluded and different camera/illuminator relationships. Starting with detecting these contours in the iconic range image, the descriptions are generated moving up a hierarchy from contour, surface, object, to scene. We use conical and cylindrical surfaces as primitives. The emphasis in this work is data-driven bottom up autonomous processing, generating object descriptions from complicated scenes without referring to specific pre-stored object models. Therefore, in the process of generating descriptions, we exploit the fact that general coherent

relationships, such as symmetry, collinearity, and having a common axis, which are present among lower-level elements in the hierarchy allow us to hypothesize upper-level elements. The analysis program has been applied to complex scenes containing cups, pans, and toy shovels.

The second piece of work, by Tomita [24], is edge-based and directed toward object recognition. A light-stripe rangefinder image is first segmented into edges and surfaces. This segmentation is done in 3D space; edges are classified as either 3D straight lines or circular curves, and surfaces are either planar or conic. An object model consists of component edges and surfaces and their interrelationships. Our model representation can accommodate not only objects with rigid, fixed shape, but also objects with articulations between their parts, such as rotational-joint or linear-slide motions. The matching process is rather straightforward. A transformation from an object model to the scene is hypothesized by initially matching a few scene features with model features. The transformation is then tested with the rest of the features for verification. The object model is constructed either interactively by using sample scenes or by deriving the model representation from the PADL-2 solid-modeling system.

During the course of our work we are accumulating facilities useful for acquiring, processing, and displaying three-dimensional range images. The collection includes: data acquisition system for industrial setting, a program library for boundary detection, 3D curve segmentation, 3D edge detection, etc. a 3D display program using fast 3D graphics to quickly see the data and model description overlaid.

4.2. From Aerial Photos: 3D Mosaic System

The 3D Mosaic system now can combine both stereo image analysis and monocular image analysis as sources of information to be accumulated into a consistent 3D description of a scene. Each view of the scene, which may be either a single image or a stereo pair, undergoes analysis which results in a 3D wireframe description that represents portions of edges and vertices of objects. The surface-based scene description is constructed from the wireframes. With each successive view, the description is incrementally updated and gradually become more accurate and complete. Task-specific knowledge, involving block-shaped objects in an urban scene, is used to extract the wireframes and construct and update the model. A complete report is included in these proceedings [5].

5. Digital Mapping and Photo Interpretation

Work in the area of Digital Mapping and Photo Interpretation [13] has continued and expanded into two new areas. The first is map-guided feature extraction, and the second area involves new research in the area of building rule based systems for photo interpretation.

5.1. Map Guided Feature Extraction

We have developed an approach to map guided image analysis using a region-based segmentation system. A full paper is included in these proceedings [14]. This segmentation system has been used to search a database of images that are in correspondence with a geodetic map to find occurrences of known buildings, roads, and natural features. The map predicts the approximate appearance and position of a feature in an image. The map also predicts the area of uncertainty caused by errors in the image to map correspondence. The segmentation process then searches for image regions that satisfy 2-dimensional shape and intensity criteria. If no initial region is found, the process attempts to merge together those regions that may satisfy these criteria. Several detailed examples of the segmentation process are given.

This work uses the CONCEPTMAP database [11] from the MAPS system [12, 15] as its source of map knowledge. In the CONCEPTMAP database, map knowledge is represented as three dimensional descriptions of man-made features, natural features, and conceptual features. Examples of man-made features are buildings, roads, and bridges; natural features are rivers, lakes, and forests, and conceptual features are political boundaries, residential neighborhoods, and business areas. These feature positions are represented in the map database in terms of $\langle \text{latitude}, \text{longitude}, \text{elevation} \rangle$.

5.2. Rule Based Interpretation of Aerial Imagery

We have begun research on the development of a rule based system, SPAM [17, 16], that uses map and domain specific knowledge to interpret airport scenes. This research investigates the use of a rule based system for the control of image processing and interpretation of results with respect to a world model, as well as the representation of the world model within an image/map database.

SPAM, A System for Photo interpretation of Airports using MAPS, is an image interpretation system. It coordinates and controls image segmentation, segmentation analysis, and the construction of a scene model. It provides several unique capabilities to bring map knowledge and collateral information to bear during all phases of the interpretation. These capabilities include:

- The use of domain dependent spatial constraints to restrict and refine hypothesis formation during analysis.
- The use of explicit camera models that allow for the projection of cartographic information onto the image.
- The use of image-independent metric models for shape, size, distance, absolute and relative position computation.
- The use of multiple image cues to verify ambiguous segmentations. We look for instances of features in previous or contemporary images to detect missing features in the model.

The task of airport image analysis has several interesting properties. First, airports are a complex organization of man-made structures placed over a large ground area. While the actual spatial arrangement of typical structures such as runways, terminal buildings, parking lots, etc. varies greatly between airports, the types of structures normally found in an airport scene are well understood. The airport task provides a "knowledge rich" environment, where functional relationships between structures preclude arbitrary spatial arrangements and provide spatial constraints. One can view the interpretation problem from the generalist standpoint, and test how rule-based interpretation can capture models of "generic" airports as well as use knowledge about the layout of specific airports in a site-specific manner.

Second, a body of literature [3, 7] on airport planning is readily accessible and provides general design constraints. Knowledge acquisition for spatial constraints therefore does not involve examination of large numbers of sample airports.

Currently the system can extract and identify some runways, taxiways, grassy areas, and buildings from region-based segmentations for several images of National Airport in Washington D.C. It builds multiple plausible functional area descriptions, which support the runway and access road portions of the scene model. It cannot currently recognize a complete airport scene that satisfies its internal model. Many problems remain, some in reliable low-level feature extraction from the

imagery, others in the design and implementation of effective recognition strategies using the rule-based approach. However, we believe that the integration of map knowledge, image processing tools, and rule-based control and recognition strategies will be shown to be a powerful computational organization for automated feature extraction from aerial imagery.

6. References

1. Baker, H. H. Depth from Edge and Intensity Based Stereo. Tech. Rept. AIM-347, Stanford Artificial Intelligence Laboratory, 1982.
2. Dew, P. M. and Chang, C. H. Passive Navigation by a Robot on the CMU Warp Machine. Tech. Rept. (forthcoming). Carnegie-Mellon University Computer Science Department, 1984.
3. Froesch, C., and Prokosch, W.. *Airport Planning*. John Wiley and Sons, Inc., New York, N. Y., 1946.
4. Henderson, R.L., Miller, W.J., and Grosch, C.B. "Automatic stereo reconstruction of man-made targets." *SPIE* 186, 6 (1979), 240-248.
5. Herman, M. and Kanade, T. The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images. In this proceedings, 1984.
6. Horn, B. K. P. and Sjoberg, R. W. "Calculating the Reflectance Map." *Applied Optics* 18 (1979), 1770-1779.
7. Horonjeff, R., and McKelvey, F. X.. *Planning and Design of Airports*. McGraw-Hill Book Company, New York, N. Y., 1983. Third Edition
8. Kanade, T. and Webb, J. Vision on a Systolic Array Machine. *Proc. Computer Architecture for Vision Workshop, DARPA*, May 2-3, 1984.
9. Lucas, B. D. and Kanade, T. Optical Navigation by the Method of Differences. In this proceedings, 1984.
10. Matthies, L. H. and Thorpe, C. E. Experience with Visual Robot Navigation. *Proceedings of IEEE: Ocean-84, IEEE*, Sept., 1984.
11. McKeown, D.M. Concept Maps. *Proceedings: DARPA Image Understanding Workshop*, Sept., 1982, pp. 142-153. Also available as Technical Report CMU-CS 83-117
12. McKeown, D.M. MAPS: The Organization of a Spatial Database System Using Imagery, Terrain, and Map Data. *Proceedings: DARPA Image Understanding Workshop*, June, 1983, pp. 105-127. Also available as Technical Report CMU-CS 83-136
13. McKeown, D. M. and Lukes, G. E. Digital Mapping and Image Understanding. *Archives of the XVth Congress on Photogrammetry and Remote Sensing. International Society for Photogrammetry and Remote Sensing, Rio de Janeiro, Brazil*, June, 1984, pp. 690-697.
14. McKeown, D.M., Denlinger, J.L. Map Guided Feature Extraction from Aerial Imagery. *Proceedings of Second IEEE Computer Society Workshop on Computer Vision: Representation*

and Control, May, 1984. Also available as Technical Report CMU-CS-84-117

15. McKeown, D.M., Digital Cartography and Photo Interpretation from a Database Viewpoint. In *New Applications of Databases*, Gargarin, G. and Golembe, E., Ed., Academic Press, New York, N. Y., 1984.

16. McKeown, D.M., Harvey, W.A. and McDermott, J. Rule Based Interpretation of Aerial Imagery. *Proceedings of IEEE Workshop on Principles of Knowledge-Based Systems*, Dec, 1984.

17. McKeown, D.M. and McDermott, J. Toward Expert Systems for Photo interpretation. *IEEE Trends and Applications '83*, May, 1983.

18. Ohta, Y. and Kanade, T. Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming. Tech. Rept. CMU-CS-83-162, Carnegie-Mellon University Computer Science Department, October, 1983.

19. Shafer, S. A. *Shadow Geometry and Occluding Contours of Generalized Cylinders*. Ph.D. Th., Carnegie-Mellon University, May 1983.

20. Shafer, S. A. Using Color to Separate Reflection Components. Tech. Rept. (in preparation), University of Rochester, May, 1984.

21. Smith, D. R. and Kanade, T. Autonomous Scene Description with Range Imagery. In this proceedings, 1984.

22. Thorpe, C. E. An Analysis of Interest Operators for FIDO. *Proceedings of IEEE Workshop on Computer Vision: Representation and Control*, IEEE, 1984.

23. Thorpe, C. E. Path Relaxation: Path Planning for a Mobile Robot. *Proceedings of the National Conference on Artificial Intelligence*, AAAI, Austin, Texas, August, 1984.

24. Tomita, F. and Kanade, T. A 3D Vision System: Generating and Matching Shape Descriptions in Range Images. *Preprints of the Second International Symposium of Robotics Research*, Kyoto, Japan, Aug. 20-23, 1984.

IMAGE UNDERSTANDING TECHNIQUES
FOR AUTONOMOUS VEHICLE NAVIGATION

Azriel Rosenfeld
Larry S. Davis
Allen M. Waxman

Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

This report summarizes research carried out during the period December 1982-August 1984 on Contract DAAK70-83-K-0018. The focus of this research is on image understanding techniques applicable to autonomous land vehicle navigation. Particular emphasis has been placed on time-varying imagery analysis, but some work has also been done on two-dimensional shape analysis and three-dimensional object recognition. A second major emphasis was the development of an approach to road network following and obstacle avoidance tasks; this work has now received major additional funding under the DARPA Autonomous Vehicle Program.

1. INTRODUCTION

The Computer Vision Laboratory of the Center for Automation Research at the University of Maryland first received support under the DARPA Image Understanding Program in 1976. This support, which was funded through the U.S. Army Night Vision and Electro-Optics Laboratory in Fort Belvoir, VA, emphasized the development of techniques for detecting tactical targets on infrared imagery.

In December 1982 a new effort, entitled "Autonomous Vehicle Navigation", was initiated at Maryland under the Image Understanding Program. This work was funded through USANVEOL under Contract DAAK70-83-K-0018, with Dr. George R. Jones as COTR and the Westinghouse Corp. as a subcontractor.

The focus of the research being conducted under this project is the development of image understanding techniques applicable to autonomous land vehicle navigation. Particular emphasis has been placed on time-varying imagery analysis, but some work has also been done on two-dimensional shape analysis and three-dimensional object recognition. A second major emphasis was the development of an approach to road network following and obstacle avoidance tasks; this work has now received major additional funding under the DARPA Autonomous Vehicle Program.

2. TIME-VARYING IMAGERY ANALYSIS

Earlier work on analysis of optical flow fields at Maryland dealt with propagation of local object motion estimates along contours and with flow field smoothing. An especially powerful smoothing technique [1] utilizes global information about the motion field, derived from the histograms of the components of the estimated motion. The method is an adaptation of the "superspike" image enhancement algorithm to motion field estimation. Experiments indicate that the method can yield more accurate and precise estimates of motion than previously proposed motion estimation algorithms.

A major new effort on optical flow field analysis was initiated when Dr. Allen M. Waxman joined the University of Maryland in May, 1983. Dr. Waxman's initial work on the structure from motion problem was done in collaboration with Shimon Ullman of MIT [2]. This work involved a new formulation and method of solution of the image flow problem. The two-dimensional image flow is generated by the relative rigid body motion of a smooth, textured object along the line of sight to a monocular camera. By analyzing this evolving image sequence, one hopes to extract the instantaneous motion (described by six degrees of freedom) and local structure (slopes and curvatures) of the object along the line of sight. The formulation relates a new local representation of an image flow to object motion and structure by twelve nonlinear, algebraic equations. The representation parameters, termed observables, are given by the two components of image velocity, three components of rate-of-strain, spin, and six independent image gradients of rate-of-strain and spin, evaluated at the point on the line of sight. These kinematic variables are motivated by the deformation of a finite element of flowing continuum. A method for solving these equations was devised and successfully implemented on a VAX computer. A number of examples were explored revealing two classes of ambiguous scenes (i.e., nonunique solutions are obtained). A sensitivity analysis was also begun in order to estimate noise levels in the representation parameters which still yield acceptable solutions; indications are that the method is quite stable. Finally, an approach is suggested by which the kinematic variables may be extracted from evolving contours in an image sequence.

Dr. Waxman has formulated a general "image flow paradigm" [3] describing the relationship between a three-dimensional (3-D) scene consisting of several objects in rigid body motion, and its associated two-dimensional (2-D) time-varying imagery. The paradigm addresses a number of theoretical issues: What is a useful representation for the 2-D flow field and how can it be obtained from the time-varying imagery? How should a flow field be segmented and how do these segmentation boundaries relate to the 3-D scene itself? How is the 3-D structure and motion of objects in the scene recovered from the 2-D flow representation and its segmentation? In attempting to answer these questions, a variety of interesting concepts arise such as image neighborhood deformation, evolving contours, space-time stream tubes, virtual contours and virtual tubes, flow analyticity, boundaries of analyticity, kinematic analysis and structure-motion compatibility. The various "elements" of the paradigm are supported by analytic techniques, some of which have already been developed, some of which are now under investigation, and others which remain to be studied.

Initial work on implementation of the paradigm has been carried out as part of the Ph.D. dissertation of Mr. Kwangyeon Wahn [4]. In the kinematic analysis of time-varying imagery, where the goal is to recover object surface structure and space motion from image flow, an appropriate representation for the flow field consists of a set of deformation parameters which describe the rate-of-change of an image neighborhood. We have developed methods for extracting these deformation parameters from evolving contours in an image sequence; the image contours being manifestations of surface texture seen in perspective projection. Our results follow directly from the analytic structure of the underlying image flow; no heuristics are imposed. The deformation parameters we seek are actually linear combinations of the Taylor series coefficients (through second derivatives) of the local image flow field. Thus, a by-product of our approach is a second-order polynomial approximation to the image flow in the neighborhood of a contour. For curved surfaces this approximation is only locally valid, but for planar surfaces it is globally valid (i.e., it is exact). Our analysis reveals an "aperture problem in the large" in which insufficient contour structure leaves the set of twelve deformation parameters under-determined. We also assess the sensitivity of our method to the simulated effects of noise in the "normal flow" around contours, as well as the angular field of view subtended by contours. The sensitivity analysis is carried out in the context of planar surfaces executing general rigid body motions in space. Future work will address the additional considerations relevant to curved surface patches.

An Image Flow Simulator for the experimental study of optical flow fields was developed as part of the M.S. thesis of Mr. Sarvajit S. Sinha [5]. The purpose of the Simulator is, from a knowledge of structure and motion, to display the 2-D image sequence and associated flow. This 3-D graphics

animation package simulates motion of objects through space and also the evolution of surface contours through time. It includes graphics algorithms for projection, clipping, hidden surface removal, shading and animation.

Another part of Mr. Sinha's M.S. thesis deals with "dynamic stereo" [6], a new concept in passive ranging to moving objects which is based on the comparison of multiple image flows. It is well known that if a static scene is viewed by an observer undergoing a known relative translation through space, then the distance to objects in the scene can be easily obtained from the measured image velocities associated with features on the objects (i.e., motion stereo). But in general, individual objects are translating and rotating at unknown rates with respect to a moving observer whose own motion may not be accurately monitored. The net effect is a complicated image flow field in which absolute range information is lost. However, if a second image flow field is produced by a camera whose motion through space differs from that of the first camera by a known amount, the range information can be recovered by subtracting the first image flow from the second. This "difference flow" must then be corrected for the known relative rotation between the two cameras, resulting in a divergent relative flow from a known focus of expansion. This passive ranging process may be termed Dynamic Stereo, the known difference in camera motions playing the role of the stereo baseline. We have developed the basic theory of this ranging process, along with some examples for simulated scenes. Potential applications are in autonomous vehicle navigation (with one fixed and one movable camera mounted on the vehicle), coordinated motions between two vehicles (each carrying one fixed camera) for passive ranging to moving targets, and in industrial robotics (with two cameras mounted on different parts of a robot arm) for intercepting moving workpieces.

In collaboration with Dr. Jacqueline LeMoigne, Dr. Waxman has also been studying the feasibility of using projected light grids to construct range maps of a robot's immediate environment. They have addressed a number of operational considerations and image processing tools relevant to this task domain, including the issues of operating in ambient lighting, smoothing of range texture, grid pattern selection, albedo normalization, grid extraction and coarse registration of image to projected grid.

A study of object tracking and occlusion analysis has been carried out in connection with the M.S. thesis of Mr. Nader Kazor [7]. A system was developed that builds a map of the environment by tracking moving objects and detecting instances of occlusion. This information is used to place bounds on the ranges and bearings of the stationary objects in the scene.

3. OTHER TOPICS

A method of recovering closed boundaries of two-dimensional shapes from disconnected boundary

segments was developed as part of the Ph.D. dissertation of Ms. Tsai-Yun Phillips, under the direction of Dr. Takashi Matsuyama of Kyoto University. A geometric labeling scheme was introduced for the Voronoi diagram of a set of straight line segments in the Euclidean plane, and a method was developed of recovering the medial axis of a closed boundary by using the labeled Voronoi diagram [8]. Algorithms were then developed for computing the labeled Voronoi diagram for a set of digital line segments, using a labeled Euclidean distance transform [9]. Finally, a digital algorithm for extracting the digital medial axis from the labeled Voronoi diagram was implemented [10].

A study of visibility in planar polygonal regions was conducted in connection with the M.S. thesis of Mr. Mark Doherty [11]. Algorithms were developed for finding minimal sets of points from which the entire region is visible. The relationship between this task and that of decomposing the region into a minimal union of star-shaped subsets was also investigated.

The use of Hough transform methods for three-dimensional object recognition was investigated as part of the Ph.D. dissertation of Ms. Teresa M. Silberberg [12]. This involved an iterative procedure in which straight line segments in the image are matched by finding the parameters of a viewing transformation of a three-dimensional model consisting of line segments. Assuming the scale of the object is known, there are three orientation and two translation parameters to be estimated. Initially a sparse, regular subset of parameters and transformations is evaluated for goodness-of-fit; then the procedure is repeated by successively subdividing the parameter space near current best estimates of peaks.

In the area of software development, an interface between the C and Franz Lisp languages was implemented and documented [13]. The documentation describes in detail how C functions may be called from Lisp and vice versa.

4. AUTONOMOUS LAND VEHICLE NAVIGATION

The goal of the research being conducted on this project is to demonstrate the roles that vision can play in navigation. Our approach to visual navigation involves segmenting the general navigation task into three levels, called long-range, intermediate-range, and short-range navigation. The general flow of control between levels is that goals flow from levels of greater abstraction to levels of lesser abstraction (long-intermediate-short) and status information, concerning the achievement of those goals, passes in the opposite direction. Each level of navigation maintains a map of (some subset of) the environment to be navigated, with the map information becoming more concrete as one moves from long down to short range navigation. Specific sensors and visual capabilities are also associated with each level of navigation; these sensors and algorithms function to maintain the correctness of the map representation at that level.

The long-range navigator is responsible for determining general paths through regions of uniform visibility/navigability. Its map is a low-resolution decomposition of the environment into regions in which:

- 1) a minimal number of landmarks (either specified in the map or acquired, dynamically, during navigation) will be visible from most points within the region (uniform visibility), and
- 2) the terrain is of uniform composition (uniform navigability).

The long-range navigator constructs a path, represented as a sequence of regions, from this map that will get the vehicle from the starting location to its goal. This path might be edited as the vehicle navigates through its environment based on information obtained at lower levels of navigation and passed up to the long-range navigator.

The principal visual capability of the long-range navigator is landmark recognition. By recognizing a sufficient number of landmarks to sufficient accuracy, the long-range navigator can compute the location of the vehicle to the accuracy required for continuing the mission. The long-range navigator is responsible for planning a sequence of landmark recognition tasks, and then controlling each of these recognition tasks. The control involves first analyzing map information to, e.g., predict the appearance of the landmark and the background against which it will likely appear from the vehicle's perspective. Secondly, the camera system, which will include an electronically controlled zoom lens and pan/tilt mechanism, must be controlled to locate and identify the landmark with sufficient angular resolution.

In summary, the inputs to the long range navigator include a coarse visibility map, visual models of landmarks, a low resolution terrain map, and a mission description. This navigator must be capable of object recognition and of path generation from a region graph. Its internal data representations include a map of regions of uniform visibility/navigability; the location of the vehicle; and a sequence of (paths through) regions.

The intermediate range navigator maintains a map which is a subset of the long-range map, but represents that subset of the environment in greater detail based on analyses of its vision system. Its navigation task is to compute a path through a region of uniform visibility/navigability by identifying what we call corridors of free space. A corridor of free space is a straight swath of navigable terrain that is not so densely populated with obstacles that the vehicle could not maneuver among them. The intermediate range navigator constructs this sequence of corridors by initially considering a straight line path of nominal width, and then perturbing this path, by increasing its width and splitting it into subsegments, to get a minimal length path of sufficient "instantaneous" navigability.

The vision capabilities required at this level of navigation are by far the most complex of all three levels. The intermediate range navigator needs to maintain a three-dimensional model of its environment to ranges far greater than can be explored using active ranging; furthermore, it requires a sophisticated object recognition and terrain analysis capability. The vision algorithms will rely heavily on both multitemporal and stereo analyses.

In summary, the intermediate range navigator's inputs include a subset of the long range map, models of known obstacles, models of gross terrain features, and a path. This navigator must be capable of general image analysis for segmentation and recognition; qualitative ranging (e.g., relative depth by occlusion; motion stereo; dynamic stereo from image flows); landmark acquisition (anomaly detection); and path (corridor) planning. Its internal data representation includes a map of corridors of free space, major obstacles and terrain features.

The short-range navigator is tasked to navigate the vehicle through a single corridor of free space by identifying with that corridor a track of safe passage. It does this based on a combination of direct ranging and multispectral analysis. Direct ranging, either from a time-of-flight laser or a specially designed structured light sensor, provides the navigator with a low resolution depth map of the terrain. Based on properties of the vehicle, the short range navigator would then identify the subspace of the immediate environment through which the vehicle can move. The analysis of the range data will be supplemented by a terrain composition analysis based on multispectral data so that the navigator can distinguish rock from vegetation from water, etc.

In summary, the inputs to the short range navigator include reports from the pilot and dead reckoning subsystems, and a corridor of free space. It must be capable of ranging (using a laser scanner or structured light system), terrain composition assessment (spectral/range classification), failsafe collision avoidance (e.g., using acoustic proximity sensors), and path planning of specific tracks. Its internal data representation includes a dynamic multi-resolution terrain map.

A specific intermediate range navigation task now being investigated is road following. A system for carrying out this task will make extensive use of domain specific knowledge, including ground topography and 3-D road models; local straightness and parallelness of road edges; slow variation in road geometry (except at intersections); and discriminability of the road from its background (e.g., road edges usually give rise to intensity edges). The initial step in road following is based on multi-resolution, multi-perspective edge detection. From an initial edge picture, obtained using a gradient operator, linear features are enhanced using local support or anisotropic motion blur. Various methods are then used to select edges of interest, including verification based

on texture continuation; consistency over scale and perspective; and generalized Hough matching. Knowledge about the road (slowly varying) and vehicle motion allow edges to be predicted in subsequent views -- in other words, guided search can be used to find edges in views from new perspectives. The model and predictions can then be corrected and updated.

The road following process also involves other intermediate range navigation tasks, such as planning for anticipated road changes; negotiating intersections and curves; getting on and off roads; and (eventually) navigating among other moving vehicles. It also involves short range tasks, including avoidance of obstacles (with the aid of range data) and constraining the vehicle's path to the road corridor; as well as long range tasks, including recognition of landmarks along the road and road network navigation.

REFERENCES

1. Hu-chen Xie, Kwangyoen Wahn, Larry S. Davis, and Azriel Rosenfeld, "Optical Flow Field Smoothing by Local Use of Global Information," CAR-TR-3, CS-TR-1274, April 1983.
2. Allen M. Waxman and Shimon Ullman, "Surface Structure and 3-D Motion from Image Flow: A Kinematic Analysis," CAR-TR-24, CS-TR-1332, October 1983.
3. Allen M. Waxman, "An Image Flow Paradigm," CAR-TR-45, CS-TR-1367, February 1984.
4. Allen M. Waxman and Kwangyoen Wahn, "Contour Evolution, Neighborhood Deformation and Global Image Flow: Planar Surfaces in Motion," CAR-TR-58, CS-TR-1394, April 1984.
5. Sarvajit S. Sinha and Allen M. Waxman, "An Image Flow Simulator," CAR-TR-71, CS-TR-1417, July 1984.
6. Allen M. Waxman and Sarvajit S. Sinha, "Dynamic Stereo: Passive Ranging to Moving Objects from Relative Image Flows," CAR-TR-74, CS-TR-1421, July 1984.
7. Nader Kazor, "Target Tracking Based Scene Analysis," CAR-TR-88, CS-TR-1437, August 1984.
8. Takashi Matsuyama and Tsai Yun Phillips, "Extracting the Medial Axis from the Voronoi Diagram of Boundary Segments: An Alternative Method for Closed Boundary Detection," CAR-TR-2, CS-TR-1261, April 1983.
9. Tsai-Yun Phillips, "The Labeled Discrete Voronoi Diagram," CAR-TR-4, TR-CS-1278, May 1983.
10. Takashi Matsuyama and Tsai-Yun Phillips, "Digital Realization of the Labeled Voronoi Diagram and its Application to Closed Boundary Detection," CAR-TR-22, CS-TR-1328, October 1983.
11. Mark F. Doherty, "Computation of Minimal Iso-vist Sets," CAR-TR-87, CS-TR-1436, August 1984.
12. Teresa M. Silberberg, Larry Davis, and David Harwood, "An Iterative Hough Procedure for Three-Dimensional Object Recognition," CAR-TR-20, CS-TR-1317, August 1983.
13. Fred P. Andresen, "The Franz Lisp - C Interface," CAR-TR-68, CS-TR-1411, June 1984.

SECTION II
TECHNICAL REPORTS
PRESENTED

PREVIOUS PAGE
IS BLANK

REAL-TIME IMAGE PROCESSING AT WESTINGHOUSE: 1964 - 1984

Bruce J. Schachter

Westinghouse Defense Center
Box 746, MS 1297
Baltimore, Maryland 21203

INTRODUCTION

The Westinghouse Image Processing Group was headed by Dr. Glenn Tisdale for nearly 20 years. Dr. Tisdale retired in August 1984. We would like to take this opportunity to describe the major products produced by the group during this period as well as its future direction.

EARLY SYSTEMS

Work in real-time image processing began at the Westinghouse Defense Center in 1968. The work was theoretical until 1970 when a software simulation of a system began.

A near real-time brassboard was completed in 1974. The brassboard filled a full rack, but could only process a 100-by-100 pixel window once per second. The locations of recognized objects were indicated via an LED array surrounding the display. The system used an analog storage tube to capture the incoming video. This proved to be a major source of noise and instability.

An advanced breadboard was completed in 1978. Its front end included an A/D converter and a 128-by-128 digital frame buffer. It extracted edges and blobs from the digital imagery at near real-time rates. In this, as well as later systems, symbology could be superimposed over the live video to cue recognized objects.

AUTO-Q-SYSTEMS

Westinghouse's AUTO-Q line of processors consists of four different models. A scene matching system (known as AUTO-MATCH) was in-

stalled at NASA-Goddard in 1979. It is used for registering LANDSAT image pairs.

An AUTO-Q I unit (figure 1) was built in 1980 and flight tested by the Army at Fort A.P. Hill in 1981. It consists of an image preprocessor and military computer, all in a 1 cubic foot box. A commercial version of this unit is now in production. It uses Multibus size boards with an Intel single board computer as host.

Two AUTO-Q II units have been built. One, delivered to the Air Force, accepts linescan imagery at a 12 megapixel per second rate. The other processes full frame video or 875-line FLIR data at a 3.3 megapixel per second rate.

AUTO-Q is composed of a custom-designed preprocessor and a general-purpose postprocessor. The preprocessor algorithms are hardwired, although certain operations may be enabled, disabled, or modified under software control. The basic tasks of the preprocessor are (1) digitizing the analog video input, (2) storing the digitized video in a frame buffer, (3) conditioning the digital video, and (4) extracting edge and blob features from the imagery. The postprocessor is basically given a list of edge vectors, blobs, and image statistics. From these, it performs some type of scene analysis. The preprocessor works at such a high rate that the postprocessor can frame integrate features and control the settings of the preprocessor via feedback loops.

A block diagram of AUTO-Q II is given in figure 2. After being digitized and stored, the video data is filtered to attenuate noise. Various filter combinations can be chosen on a frame-by-frame basis. The filtered image is then passed through a programmable contrast enhancer.

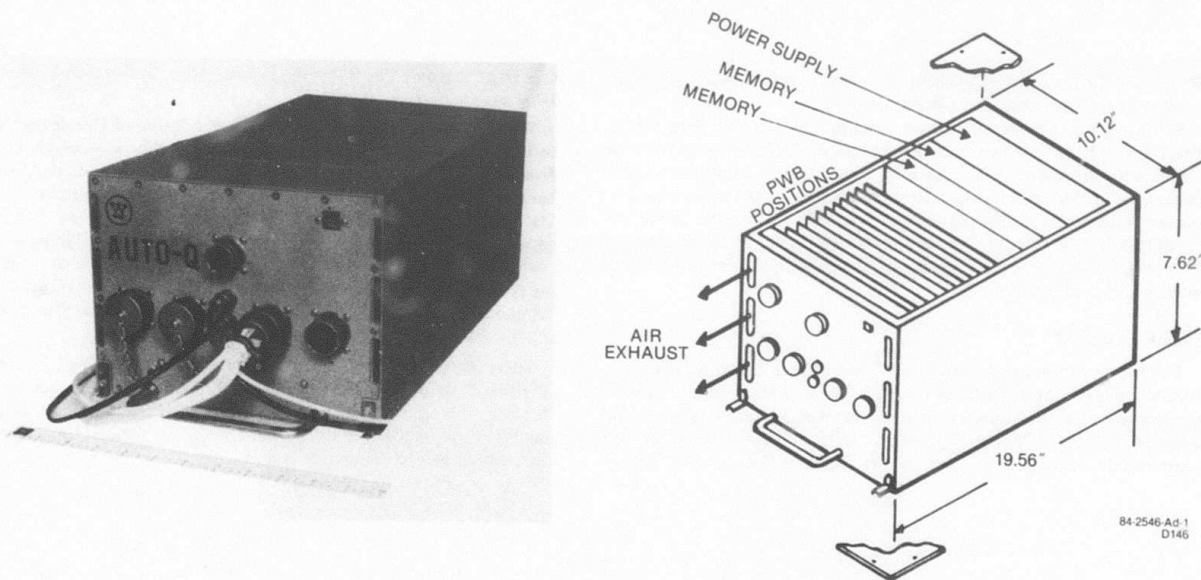
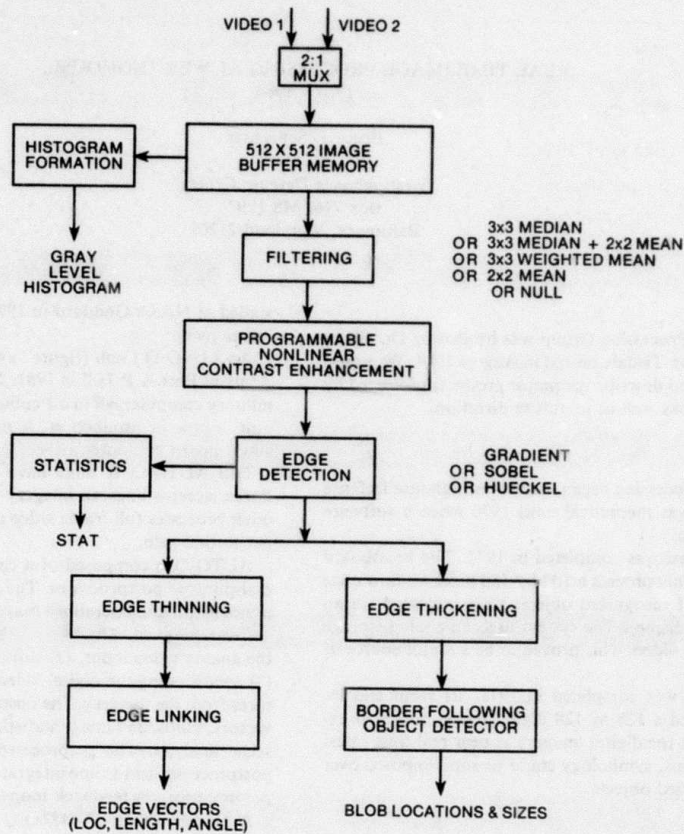


Figure 1. AUTO-Q Preprocessor and Postprocessor are Housed in a 1 Cubic Foot Box



PIPELINED ARCHITECTURE OF AUTO-Q PREPROCESSOR



ALL STAGES UNDER SOFTWARE CONTROL FRAME-BY-FRAME

Figure 2. Block Diagram of AUTO-Q Preprocessor

One of three types of edge detectors (Roberts, Sobel, or Hueckel) is then applied to the smoothed and filtered imagery.

Subsequent interest is focused on those pixels with high edge values. Two types of feature detectors are applied in parallel. One uses a thinning and linking process to construct long edge vectors. Simultaneously, a thickening operation is applied to the edge image. Blobs are then extracted from the thickened edge image by a border following operation. Up to 2000 edge vectors of various lengths and 256 blobs can be extracted for each processed frame. The processing rates of the different AUTO-Q models are given in figure 3.

VHSIC SYSTEMS

The future of image processing at Westinghouse lies with submicron VHSIC boards - both specially designed image processors and generic signal processors. Multispectral, multisensor image processors will be the norm in the 1990's. Meanwhile, older models will continue to go into commercial production as part of the Defense Center's Technology

Transfer Program. The following information is taken from an *Aviation Week* article of July 30, 1984.

Westinghouse's Phase 1 VHSIC chip set consists of five devices which serve as the building blocks for hierarchical multiprocessor systems. The five chips are Pipeline Arithmetic Unit, Extended Arithmetic Unit, Enhanced Arithmetic Unit Multiplier, General Purpose Controller, and 64K Static RAM. The Westinghouse team is the only one of the six Phase 1 contractors to use Complementary Metal-Oxide Semiconductor (CMOS) technology for all members of its chip set - a technology that is the leading contender for the follow-on Phase 2 effort. One VHSIC brassboard being built under the Phase 1 contract is an electro-optical image processor for the M1 tank.

There is growing confidence that the much more challenging objectives of Phase 2 calling for 0.5 micron feature size and a further 100-fold increase in throughput can be achieved in the VHSIC Phase 2 effort. It is our belief that processing rates will be sufficient for real-time image understanding before the decade is over.

Name	Technology	Year	Number of Units	Ops/s	512 × 512 Frames/s	Customers
AUTO-Q I	MSI TTL	1980	45	400 Million	3	Army, AF, NASA MIT, SU, DEC, etc.
AUTO-Q II	MSI TTL	1983	2	1 Billion	13	AF, Army
VHSIC ϕ 1	1.25 μ m CMOS	1985	*	*	*	Army
VHSIC ϕ 2	0.5 μ m CMOS	1988	*	*	*	Proposal Submitted

*Stars Denote Information Cannot Be Published at this Time

Figure 3. Vital Statistics of Westinghouse's Image Processors

VLSI IMPLEMENTATION OF SYSTOLIC AND 3-D CELLULAR ARCHITECTURES FOR IMAGE PROCESSING

J. Greg Nash, R. D. Etchells, J. Grinberg, S. Hansen, M. J. Little, G. R. Nudd, K. Petrozolin, R. Turk

Hughes Research Laboratories
3011 Malibu Canyon Rd.
Malibu, CA 90265

Abstract

We report here on several implementations of computing architectures for use in array-based 2-D image processing applications. First we will describe our linear systolic array, which has been built using a custom designed VLSI Multiplication Oriented Processor (MOP) chip as the processing element. It is capable of performing DFTs, 1-D and 2-D convolutions, and solving Toeplitz linear systems. A second, more general systolic array we are building, is based on the Faddeev algorithm. This machine, which is ultimately intended to be a general linear algebraic processor, is presently capable of matrix operations, linear system solution, matrix factorization, and least squares solutions. Finally, we will briefly describe our 3-D computer, which combines wafer scale integration, bus lines through wafers, and inter-wafer connections, to provide a computing capability with several orders of magnitude combined improvement in power dissipation, throughput, size and cost, when compared to present special purpose computers.

Introduction

In this paper we describe three computing architectures, which vary from very special purpose with a relatively simple implementation to more general purpose (in terms of image understanding) with a complex implementation. In Section I we discuss our linear systolic array and give an example of how it can be used to solve a special type of linear system in $O(n)$ time steps using only $O(n)$ processing elements (PEs). In Section II we describe a 2-D array of PEs capable of much more general operations in $O(n)$ time steps using an $O(n)$ array of PEs. Most of the discussion will center on the Faddeev algorithm, since it is the architectural basis of this array. Finally, we will briefly describe our 3-D computer and summarize its capabilities in Section III.

I. Linear Systolic Array

The linear systolic array architecture has been recognized as a relatively inexpensive approach to providing increased throughput in a number of important application areas. [1] Such arrays are easily expandable, relatively simple to design and interface, while offering a surprisingly large range of calculational capabilities. We describe here a design that uses a custom VLSI chip as a PE

which has been built in an optimum way for linear systolic arrays. In addition we have integrated on to the chip a special high speed multiplier [2] which is far more area-time efficient than conventional fast parallel multipliers.

Architecture

Our prototype system, shown in Figure 1, consists of nine replicated copies of our custom MOP chip, plus a commercial divider. (Each MOP chip serves as a PE.) Eight of these MOP chips are used in the systolic array and one (MOP AUX) serves as a fast buffer memory for the rest of the array. Since many algorithms for linear systolic arrays result in only alternate processors active at any time, we effectively time multiplex operations so that all processors can be active all the time. For example the eight MOP chips can be used to solve a 16th order Toeplitz linear system.

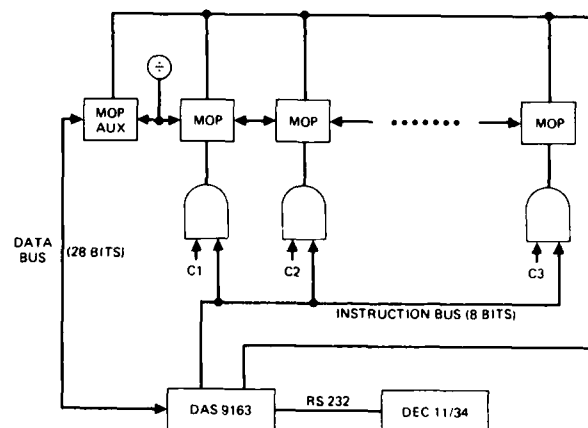


Figure 1. System block diagram of linear systolic array. The inputs C_i to the MOP chips are used to gate control signals to each chip.

Instructions are broadcast to all PEs by our control unit (Tektronix 9163 system). Each PE has a gate associated with it to control receipt of instructions. In some "wavefront" type algorithm implementations this is necessary to prevent PE's which are not part of the wavefront from receiving

instructions. As can be seen in Figure 2, there is one PE per board. All the boards are plugged into alternate slots in a VME bus. This organization provides simple diagnostic means for debugging the operation of this prototype linear array.

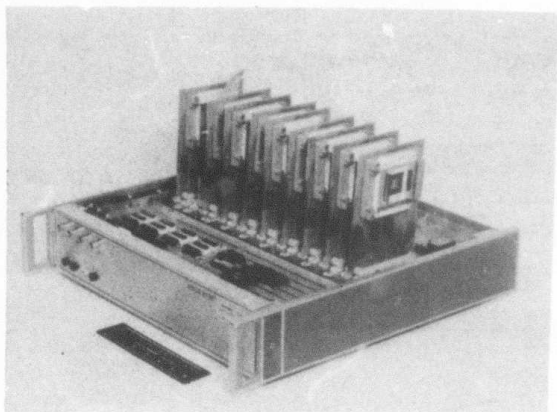


Figure 2. Photograph of linear systolic array. Each board is a PE (one MOP chip). Most of the TTL circuitry is used to implement and interface the divide function, which is performed by commercial chip.

We have written an assembler along with code to download programs from our 11/34 host computer. There is presently an RS-232 link to our host computer, which will be replaced by a more direct link in the near future.

Multiplication Oriented Processor Chip

Considering the large number of applications of systolic type processor arrays, it would be difficult to design a single, generic custom chip containing the entire spectrum of desirable attributes without significant sacrifices in performance and chip area. As an alternative approach, we have used a design methodology which offers a generic type of chip that can be easily designed to fit a specific application or range of applications. This generic capability is a result of a number of features associated with the way the chip is organized. In this section we discuss the MOP chip as an example of such a generic approach to providing PEs for systolic arrays.

In order to provide the necessary flexibility for data movement within and between chips, we selected a bus oriented and bit slice architecture. Processor arrays generally need large data bandwidths and flexibility in I/O capabilities. To support these needs we have provided the MOP with two bidirectional, tri-state, parallel I/O ports. Each of these has two output latches and two input latches and can send or receive two words per clock cycle.

The speed of the MOP chip has been enhanced by a number of features. First, it uses dual buses so that two operands can be transferred per clock cycle. All arithmetic algorithms have the necessary control embedded in hardware. This adds considerable speed over a microcoded approach and eases the controller design problem. The multiplier (and future divide and square root) circuitry runs on its own set of high speed clocks that are separate, but in synchronism with, the slower system clocks. The multiplier uses a radix-4 algorithm for an additional factor of two speed-up over conventional algorithms and, finally, has a three stage pipeline to allow very high partial product accumulation speeds.

The MOP chip, shown in Figure 3, also has 10 general purpose, dual port, 28-bit registers, two 16-word deep LIFO stacks, and a Manchester₂ type ripple adder. The chip size is 290x250 mils², and power consumption is 0.75 to 2W, depending on processing parameters. The system clocks were designed for 4MHz operation and the high speed multiplier clocks are intended to be run at 16MHz.

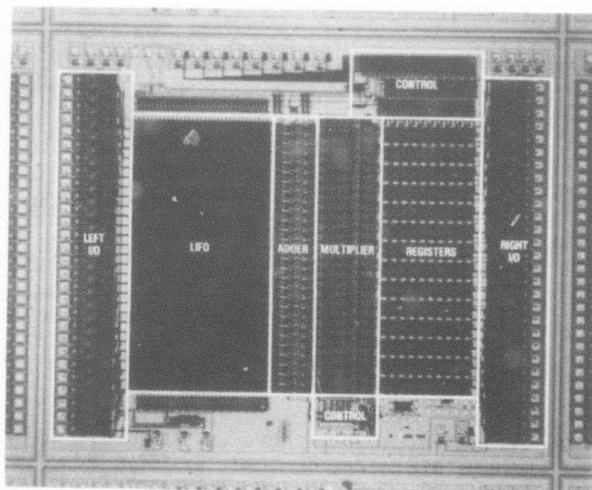


Figure 3. Photograph of MOP chip showing important regions.

Systolic Implementation of Toeplitz System Solution

We have looked at a number of algorithms that can be implemented on a linear systolic array (e.g., splines, 1-D and 2-D convolutions). We describe here briefly one of these: a Toeplitz linear system,

$$H X = Y \quad (1)$$

where H is a Toeplitz matrix. This system can be solved by using the Levinson algorithm [3,4]

$$X = U^{-1} D [U^+]^{-1} Y$$

where U is an upper triangular matrix and D a

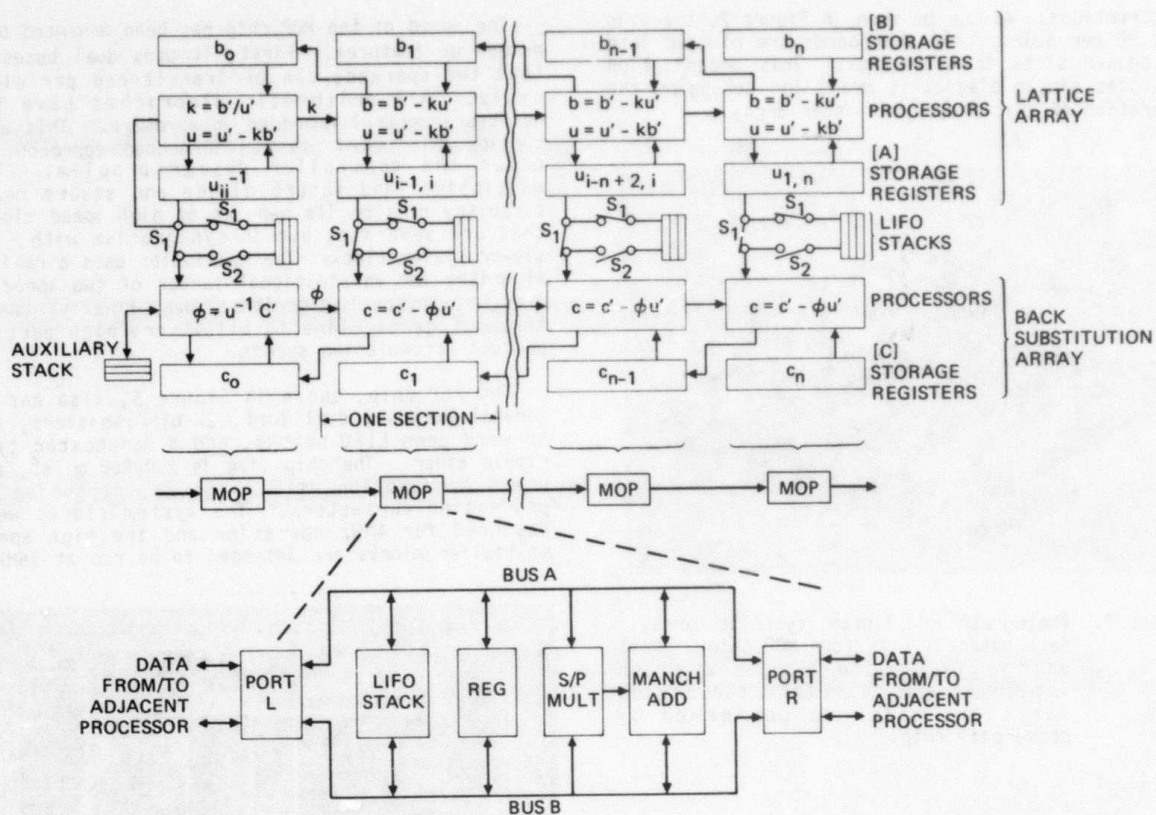


Figure 4. Diagram of linear systolic array system (top) and functional layout of MOP chip (bottom). The lattice array generates the elements of U. Switches S_1 are closed and S_2 open during the lattice array and first backsubstitution operations. During the second backsubstitution operation S_2 is closed and S_1 is open. Calculations performed by the processors are indicated in their respective boxes.

diagonal matrix with elements $u_{11}, u_{22}, \dots, u_{nn}$. To perform this operation on our systolic array the following successive steps are necessary:

Generation of U^t

First Back Substitution $A = [U^t]^{-1}C$

Second Back Substitution $X = U^{-1}A$.

The essential arithmetic elements in this process are high speed multiplication and a LIFO data reflection to perform the transpose. The linear array configuration for this is shown in Figure 4. This array provides $O(n)$ speed using $O(n)$ PEs. A complete solution to a 16th order Toeplitz system requires approximately 175 μ sec.

II. Linear Algebraic Processor

One approach to image analysis has been to relate such problems to specific concepts in linear algebra and matrix theory. [5] Therefore, a concurrent processor capable of performing a general class of linear algebraic operations would

be of great value here. However, one of the principal problems encountered in designing concurrent computing systems is that of providing a sufficiently general range of capabilities without undue addition of hardware and interconnection requirements. What we describe here is an algorithm based on that of Faddeev [6,7] which offers a systematized means for performing a variety of matrix operations. It can be adapted to use on existing 2-D concurrent computing arrays or as the basis for a computing architecture in itself.

We will first describe the Faddeev algorithm, highlighting its advantages and deficiencies for performing matrix computations. We will then show how it can be modified to provide a wider range of capabilities with improved numerical stability. Finally we will discuss its application to an important set of problems (least squares).

Faddeev Algorithm

To illustrate the Faddeev algorithm we consider the simple case of finding

$$c_1x_1 + c_2x_2 + \dots + c_nx_n + d,$$

where c_1, c_2, \dots, c_n and d are given numbers, and x_1, x_2, \dots, x_n is the solution to the linear system of equations

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2$$

$$\dots$$

$$a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n$$

whose determinant is non-zero. The problem can be codified by writing it as

$$\begin{array}{ccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \\ \hline -c_1 & -c_2 & \dots & -c_n & d \end{array}$$

or in abbreviated form,

$$\begin{array}{c|c} A & B \\ \hline -C & d \end{array}, \quad (2)$$

where B is a column vector and C is a row vector. If a suitable linear combination of the rows above the line (from A and B) are added to the row beneath the line (e.g., $-C+WA$ and $d+WB$, where W specifies the appropriate linear combination), so that only zeroes appear in the lower left hand quadrant, then the desired result, $CX+d$, will appear in lower right hand quadrant. This follows because the annulment of the lower left hand quadrant requires that

$$W=CA^{-1},$$

so that

$$d+WB=d+CA^{-1}B.$$

Since $X=A^{-1}B$, we have the final result

$$d+WB=d+CX.$$

The simplicity of the algorithm is due to the absence of a necessity to actually identify multipliers of the rows of A and the elements of B ; it is only necessary to "annul the last row." This can be done by ordinary Gaussian elimination. An important feature of this algorithm is that it avoids the usual backsubstitution or solution to the triangular linear system and obtains the values of the unknowns directly at the end of the forward course of the computation, resulting in a considerable savings in added processing and storage. Statistical studies we have done show

that numerical accuracy is comparable to the usual LU decomposition and backsubstitution.

This result can be generalized to the case of rectangular matrices C, D , and B , or

$$\begin{array}{c|c} A & B \\ \hline -C & D \end{array}.$$

After the lower left hand quadrant is annulled, the result $CA^{-1}B+D$ will appear in the lower right hand quadrant. As shown in Figure 5, numerous matrix operations are possible by selective entries in the four quadrants.

POSSIBLE OPERATIONS

$$\begin{array}{c|c} A & I \\ \hline -I & O \end{array} \Rightarrow A^{-1}$$

$$\begin{array}{c|c} I & B \\ \hline -C & O \end{array} \Rightarrow CB$$

$$\begin{array}{c|c} I & B \\ \hline -C & D \end{array} \Rightarrow D + CB$$

$$\begin{array}{c|c} A & B \\ \hline -I & O \end{array} \Rightarrow A^{-1}B$$

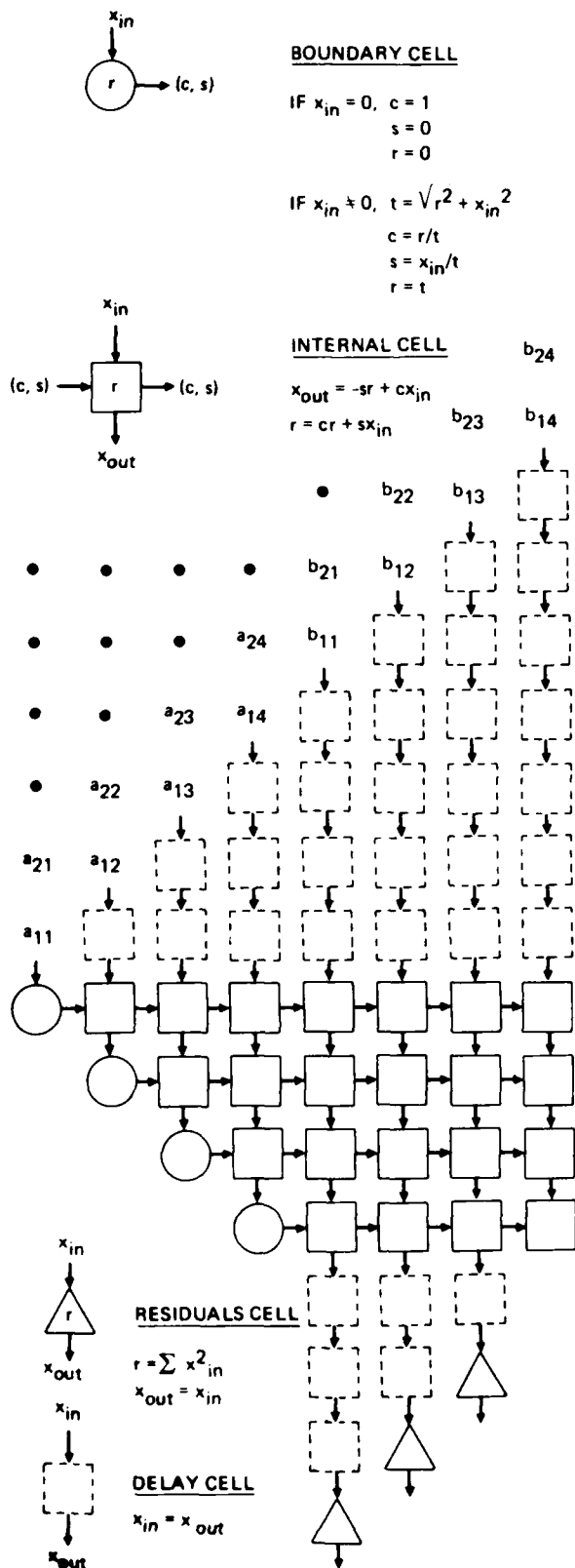
$$\begin{array}{c|c} A & B \\ \hline C & D \end{array} \Rightarrow CA^{-1}B + D$$

Figure 5. Illustration of possible matrix operations using Faddeev algorithm.

Modified Faddeev Algorithm

Although the Faddeev algorithm has some very desirable features, we would like to add an orthogonal factorization capability for added numerical stability and to permit the coefficient matrix to be non-square for over- and underdetermined systems of equations. Unfortunately, when Givens rotations are applied to the matrix

$$\begin{array}{c|c} A & B \\ \hline -C & D \end{array}$$



in the usual way (beginning in the lower left hand corner) to annul C, the result is

$$\begin{array}{c|c} \begin{bmatrix} R \\ 0 \end{bmatrix} & \begin{bmatrix} Q_1^t B \\ Q_2^t B \end{bmatrix} \\ \hline -YC + WA & YD + WB \end{array} \quad \begin{matrix} n & m-n \end{matrix}$$

$Q = [Q_1 : Q_2]$

where A is $m \times n$, R is upper triangular, Q is orthogonal, and Y is a matrix resulting from rotations on elements of C. Since the lower left hand quadrant results in $W = YCA^{-1}$, we find that the lower right hand quadrant becomes $Y(D+CX)$. Therefore, the mixing of the rows beneath the line during the rotation process causes the incorrect result to appear. For this reason it is necessary to divide the process of annulling the lower left hand quadrant into a two step procedure. First A is triangularized by Givens rotations (simultaneously applied to B); after this is completed the remainder of the process can be accomplished by Gaussian elimination using the diagonal elements of R, all of which must be non-zero if A is full rank, as pivot elements. In other words after the first step (Givens rotations) we obtain

$$\begin{array}{c|c} \begin{bmatrix} R \\ 0 \end{bmatrix} & \begin{bmatrix} Q_1^t B \\ Q_2^t B \end{bmatrix} \\ \hline C & D \end{array}$$

and after the second step, Gaussian elimination, the final result is

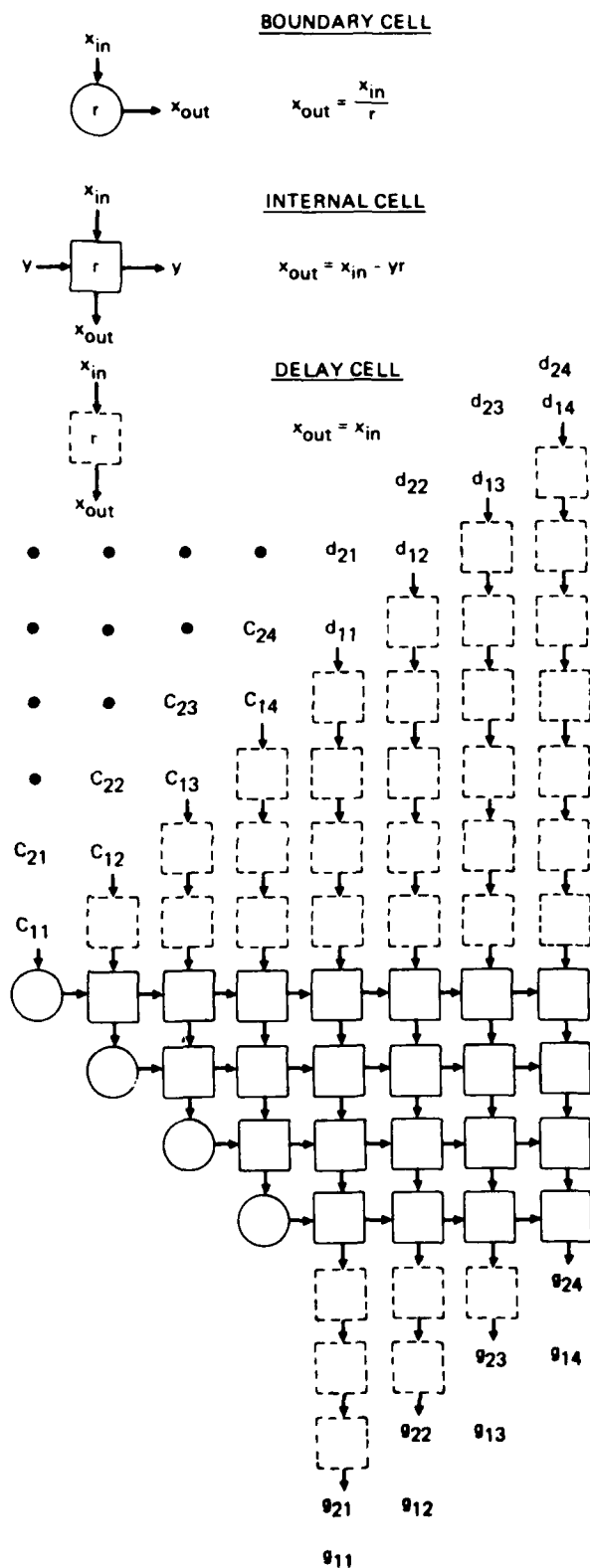
$$\begin{array}{c|c} \begin{bmatrix} R \\ 0 \end{bmatrix} & \begin{bmatrix} Q_1^t B \\ Q_2^t B \end{bmatrix} \\ \hline 0 & C R^{-1} Q_1^t B + D \end{array}$$

There are no restrictions here on the coefficient matrix A other than it be full rank. Thus, as will be seen later, least squares problems can be solved in addition to the matrix manipulation capabilities shown in Figure 5.

Architectural Considerations

As mentioned in the description of the Faddeev algorithm, one of its nice properties is that it maps well into an array architecture with data flowing in nearest neighbor fashion. One approach would be to use a triangular array [8] for this purpose. In order to correctly process the B matrix it is only necessary to extend the triangular matrix in the eastward direction as shown in Figure 6. By passing A and B down through this array with delays as shown, and performing the

Figure 6. Processor flow diagram and PE arrangement for first step (orthogonal triangularization) of modified Faddeev algorithm.



computations indicated in the circles and boxes, R and Q₁B will be left stored in the array of PEs. (In this example both A and B are mx4 in size.)

The second step in the modified Faddeev algorithm could be accomplished as shown in Figure 7. Here C and D, each of which is nx4, are also passed down through the array of processing elements in a similar way. In this case the set of operations performed in each PE is slightly different as shown. The PEs indicated by the circles each zero one column of C by pivoting on the diagonal elements of R. In this case after O(n) time steps the result CR⁻¹Q₁B+D will appear row by row coming out of the array at the bottom right.

The triangular structure of Figures 6 and 7 can be easily transformed to a square organization for more efficient implementation and generalization to handle arbitrary sized matrices.

Least Squares Problem

As an example of the use of the Faddeev algorithm we will show how it can be applied to the least squares problem [9] of finding the value of x that minimizes

$$\|Ax - b\|.$$

The usual procedure is to perform an orthogonal triangularization of the mxn (m>n) matrix A, which for the overdetermined case leads to

$$Q^T A = \begin{bmatrix} R \\ 0 \end{bmatrix} \quad (3)$$

so that

$$\|Ax - b\| = \|Q^T Ax - Q^T b\|$$

or

$$\|Ax - b\| = \|Rx - Q_1^T b\| + \|Q_2^T b\|.$$

The minimum value of $\|Ax - b\|$ is obtained with x as the solution to $Rx = Q_1^T b$. The residual is then $Q_2^T b$. These results can be found using the modified Faddeev algorithm with the data arrangement

$$\begin{array}{c|c} A & b \\ \hline -I & 0 \end{array}.$$

For example the processor arrangement shown in Figures 6 and 7 would be suitable for computing the least squares solution to $\min \|Ax - b\|$ for $i=1,2,3,4$, where A is mx4. Note, as shown in Figure 6, the residuals are very simply obtained by

Figure 7. Processor flow diagram and PE arrangement for second step (Gaussian elimination). The output matrix is $G=CR^{-1}Q_1^T B+D$.

accumulation of the squares of the first $m-4$ non-zero outputs (corresponding to the elements of $Q_2^t b$) of each of the columns associated with b_i .

For the underdetermined system, where A is $m \times n$ with $m < n$, we find after factoring that

$$\|Ax - b\| = \|[R:0]Q^t x - b\|$$

If we let

$$Q^t x = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

then y_1 can be found from the solution to the triangular system

$$Ry_1 = b$$

and y_2 is arbitrary. The usual procedure is to set $y_2 = 0$, which corresponds to taking the minimum norm solution. The underdetermined case requires that Q be applied after the solution for y_1 , so that the rotations must be accumulated during the course of the computation. This problem can be solved using the modified Faddeev algorithm with the data entries

$$\begin{array}{c|c} A^t & I \\ \hline -b^t & 0 \end{array}$$

At the end of the calculation the entries will be

$$\begin{array}{c|c} R^t & \\ \hline 0 & Q^t \\ \hline 0 & [y_1 : 0] Q^t \end{array}$$

where the desired result x^t is in the lower right hand quadrant. Of course multiple underdetermined least squares calculations, $\min \|Ax_i - b_i\|$, for $i=1,2,\dots,n$, could be performed with the entries

$$\begin{array}{c|c} A & I \\ \hline -B & 0 \end{array}$$

where $B = [b_1, b_2, \dots, b_n]^t$ contains the set of right hand side vectors. From an architectural point of view no extra PEs are necessary when processing more than one right hand side vector.

III. 3-D Computer

The 3-D computer concept consists of a large number (10^4 to 10^6) of parallel computing channels [10]. Typically the number of channels is

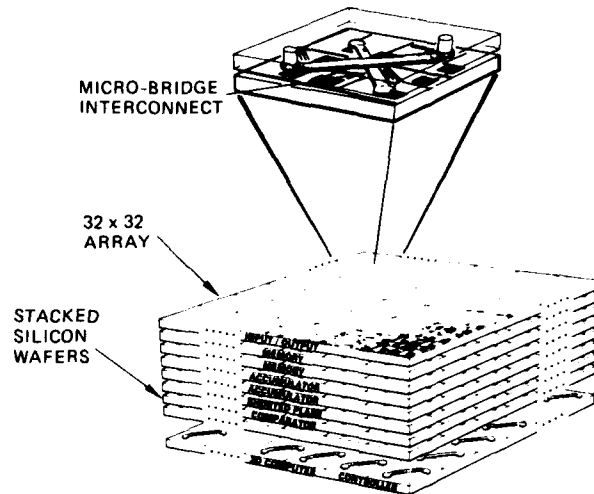


Figure 8. Basic structure of the 3-D computer.

equal to the number of pixels in the information plane, allowing the assignment of a complete processor to each pixel. This organization is in direct correspondence with (and is therefore more efficient for processing) 2-D arrayed data. However, it should be emphasized that the application of such a machine is not limited to two-dimensional data, and a very wide variety of computationally intensive applications can be performed by the structure with considerable advantage.

The 3-D parallel processor structure can be visualized as a stack of large silicon wafers lying on top of each other like a stack of coins (Figures 8 and 9). Each wafer is divided into an $N \times N$ array of primitive computing cells. The signal is transferred through the wafer at each cell and then interconnected to the corresponding cell on the adjacent wafer. In this way, $N \times N$ signal paths (bus lines) penetrate the stack of wafers. Each of these $N \times N$ data lines serves as the main data path of a primitive serial computer.

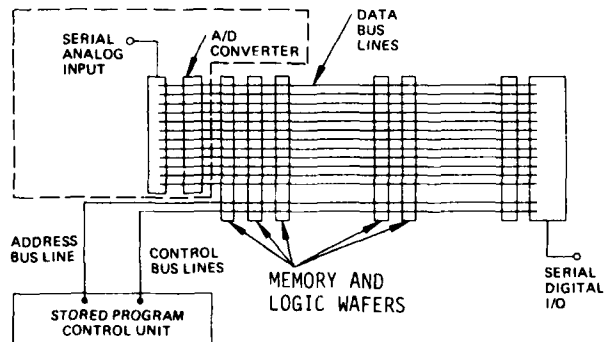


Figure 9. Organization of 3-D computer as shown from "side" view.

Functional units of each computer are arranged along these serial data buses. Each wafer in the stack contains an $N \times N$ array of computing elements of one type (such as memories, accumulators, as shown in Table 1), one such element for each of the $N \times N$ data lines. The idea is to put a stack of these primitive computing elements behind each pixel or matrix element, thereby providing a simple computer for each element of the incoming data structure.

Table 1. List of Cell Types in the 3-D Computer.

CELL TYPE	FUNCTION
Memory	Store, shift, invert/non invert, "OR," full word/MSB only, destructive/non destructive read out
Accumulator	Store, add, full word/MSB only, destructive/non destructive read-out
Replicator Plane	I/O, X/XY short, stack/control unit communication
Counter	Count in/shift out
Comparator	Store (reference), greater/equal/lower

Due to the enormous number of processing elements that must be contained on a single silicon wafer, the actual area available to each such element is quite small, presently on the order of 20 mils x 20 mils. This places a strict limitation on the number of components available with which to construct the elements. This restriction is overcome by dispersing the various functions of each computer vertically throughout a stack of wafers. In a typical example, the array may consist of 256 rows and 256 columns, for a total of 65,536 identical computers. Each of these computers would have its functions distributed over a vertical column extending through 20 or more wafers.

In the usual mode of operation, only two of the wafers in the stack will be simultaneously active, one functioning as the source of the data being processed, the other performing the processing and acting as the repository of the results of that computation. Although serial arithmetic means that the individual cellular computers are fairly slow, the massive parallelism of the array results in an enormous overall processing power. Benefits of the serial arithmetic include the simplicity of the processing elements and the fact that the processing occurs simultaneously with the memory access. As a result, the memory bandwidth of the 3-D computer is always matched to the processor bandwidth.

Numerous processing algorithms have been simulated on the 3-D architecture, including object identification and analysis, cueing routines, radar

range-Doppler computation, spotlight SAR, and matrix computations such as matrix inversion and multiplication. In Table 2 we summarize computation times for some primitive operations.

Table 2. Processing times for various primitive operations (10-MHz clock)

OPERATION	TIME
Data move (MEM \rightarrow MEM)	1.8 μ s
ADD (ACC + MEM \rightarrow MEM)	1.8 μ s
MULTIPLY (ACC x MEM \rightarrow MEM)	42.2 μ s
DIVIDE (ACC : MEM \rightarrow ACC)	127.1 μ s
SQUARE ROOT ($\sqrt{\text{ACC}}$ \rightarrow ACC)	152.6 μ s
Sobel edge operator	54.3 μ s
256 x 256 matrix multiply	12.0 ms
256 x 256, 8-bit histogram 2	1.7 ms
256 x 256 matrix inversion	10.2 ms

Redundancy

Our approach here for a prototype machine is to provide a full 2:1 redundancy on every wafer plane. In this case, there will be two computing cells, four feedthroughs, and four interwafer contacts for each pixel element in the image. Assuming 0.99 yield on a 12 x 12 mil unit cell, we have computed the yield on VLSI arrays of different size both with and without 2:1 redundancy and disconnect-type repairs. The result is shown in Table 3. Without the redundancy, the yield on a 450 x 450 mil array would be practically zero. But with 2:1 redundancy and disconnect-type repairs, the yield stays reasonable up to a 1-in. chip. The assumption in this case is that, if both cells on any pixel are defective, the chip will be thrown away. If one is willing to make 10 or fewer discrete wirings on a wafer, which means being willing to repair a wafer with 10 or fewer defective cell pairs, one can substitute a good working cell from a next neighbor pixel that has both cells good for the two defective cells. In this case, a 93% yield can be achieved on a 3.6-in. array.

Table 3. VLSI yield comparison.

ARRAY SIZE (MILS)	NO REDUNDANCY	2:1 REDUNDANCY & DISCONNECT	2:1 REDUNDANCY AND 10 SIMPLE DISCONNECT
		REPAIR ONLY	-CONNECT REPAIRS
225 x 225	0.076	0.957	1
450 x 450	0.00003	0.903	1
900 x 900	0	0.664	1
1800 x 1800	0	0.194	1
3600 x 3600	0	0.0014	0.93

System Cost

The projected cost of the 3-D computer is low. Most of the cost for present computers is for building the computer: printed boards, wiring, etc. In the 3-D computer, most of these costs are eliminated. It is not necessary to dice, bond, or package the chips; fabricate the printed boards; solder the packages onto the printed boards; or wire the printed board connectors together. The 3-D assembler simply places the wafers one on top of the other. The majority of the fabrication time will be spent on testing, and even that will be a minor effort compared to that encountered with more conventional architectures.

In summary, the 3-D computer offers several important features:

- o Very high data throughput ($>10^{10}$ instructions/sec)
- o Very low power (<30 W)
- o Extremely small size (<6 in.³)
- o Potentially low cost.

These specifications were calculated assuming an array of 128×128 cells using presently available 1:1 photolithography techniques.

Acknowledgements

The support by NSF Grant No. ECS 8016581, Office of Naval Research Contract No. N00014-81-K-0191, and DARPA and NVEOL contract DAAK-70-78-C-0163 is gratefully acknowledged.

References

1. H. T. Kung, "On the Implementation and Use of Systolic Array Processors," Proc. Int. Conf. on Computer Design, 1983, pp 370-373.
2. M. D. Ercegovac and J. G. Nash, "An area-time efficient VLSI design of a radix-4 multiplier," Proc. 1983 IEEE Conf. on Computer Design, Port Chester, N.Y., Oct. 31-Nov. 3, 1983.
3. S. Y. Kung and Y. H. Hu, "A highly concurrent algorithm and pipelined architecture for solving Toeplitz systems," IEEE Trans. on ASSP, Vol. 31, No. 1, Feb. 1983, pp. 66-76.
4. J. G. Nash, G. R. Nudd and S. Hansen, "Concurrent VLSI architectures for Toeplitz linear systems solution," presented at Gov't Microcircuit Applications Conf., Orlando, Fla., Nov. 2-4, 1982.
5. H. C. Andrews and B. R. Hunt, "Digital Image Restoration," Prentice Hall, 1977.
6. V.N. Faddeeva, Computational Methods of Linear Algebra, translated by Curtis D. Benster (Dover Publications, 1959)
7. J.G. Nash and S. Hansen, "Modified Faddeev Algorithm for Matrix Manipulation," Proc. SPIE, San Diego, CA, 1984.
8. H. T. Kung, "Systolic Array for Orthogonal Triangularization," Proc. SPIE, San Diego, CA, 1981.
9. Gene H. Golub and Charles F. Van Loan, Matrix Computations, Johns Hopkins Press, 1983.
10. J. Grinberg, G. Nudd, and R. D. Etchells, "A Cellular VLSI Architecture," Computer, Jan. 1984, p 69-81.

Semantic Network Array Processor and Its Applications to Image Understanding

V. DIXIT AND D. I. MOLDOVAN

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089

Abstract

This paper describes the organization and operation of a semantic network array processor (SNAP). The architecture consists of an array of identical cells each containing a content addressable memory, microprogram control and communication unit. The applications discussed in this paper are discrete relaxation and dynamic programming for stereo.

1. Introduction

The nature of symbolic processing used in artificial intelligence (AI) is different in many ways from conventional programming language processing. Consequently, the architecture of computers intended for AI applications should be different from today's commonly used von Neumann computers. The mapping of AI algorithms into architectures cannot be done with the same efficiency as that of numerical signal processing algorithms (mapping into systolic arrays, for example). Communication networks supporting packet switching and complicated data transfer protocols are necessary.

The vision algorithms range from very low level number crunching to symbolic processing. It is not possible to efficiently implement all these algorithms on a single machine. SNAP (Semantic Network Array Processor) currently under study at USC, addresses the high end of the vision processing. In this paper we show how SNAP can be effectively used for discrete relaxation and dynamic programming for stereo. The interested reader should see [8] for other symbolic processing applications such as pattern search, inference, and production systems.

In this paper we first present briefly the architecture and the instruction of SNAP, and then show how discrete relaxation and dynamic programming can be

processed on SNAP.

2. SNAP Architecture

The architecture consists of a square array of identical processing cells which are interconnected both globally and locally as shown in Figure 2-1.

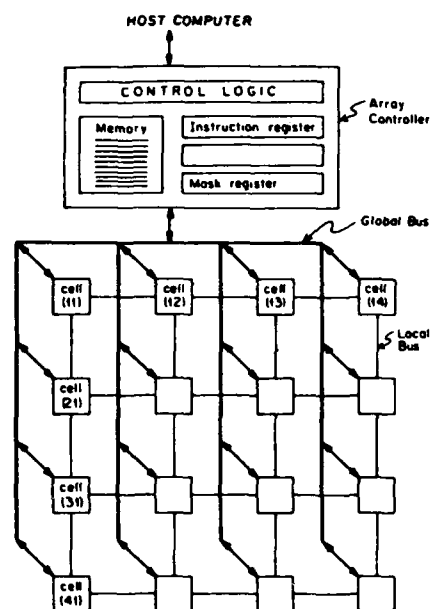


Figure 2-1: Architecture of SNAP

Its functionality rests upon two underlying concepts: associative processing and cellular array processing. Each cell contains memory control logic and communication logic. As a whole, the array is operated by an outside controller which also provides an interface between SNAP and a host computer. Our intent was to minimize the role of the global functions which affect the entire array and to provide more operational freedom for each individual cell. The cells can be microprogrammed so they can operate independently. The signals involved in the inter-cell communications are propagated from a cell

to another neighboring cell via local buses. As a result, any cell can communicate through a chain of intermediate cells to any other cell in the array. A cell's address is specified by its row number followed by its column number. Information in a particular cell can be retrieved either by its content (as in associative memories) or by that cell's address. The associative processing concept alone is not sufficient for designing efficient architectural structures for AI. This is because retrieval operations in AI are more complex than simple words; most frequently we need to match subgraphs or other patterns. Also, we need to pursue several hypotheses in parallel, and this is not possible with simple associative processors. This architecture blends the power of associative memories for performing fast information retrievals with the capability of cellular array to process various tasks in parallel. The regularity of the array is an important feature which makes feasible its VLSI implementation.

2.1. Architecture of the Cell

Each cell contains a content addressable memory (CAM), processing unit (PU), and a communication unit (CU) as shown in Figure 2-2. The cell also has some general purpose registers and flags. Processing unit has simple microinstructions such as AND, OR, NOT, RESET (a flag), MASK (a field of a string), MATCH (a string). The role of the CU is to perform data transfers between cells. The data is transferred in packets via local connections. The CU has an input queue which will hold the packet until it can claim the attention of its neighbor. Exact route taken by a packet to reach its destination depends on the routing algorithm and the actual traffic. The CU also provides access to the cell via the global bus. The global bus is used mainly by the host computer for instruction and data broadcast and data retrieval.

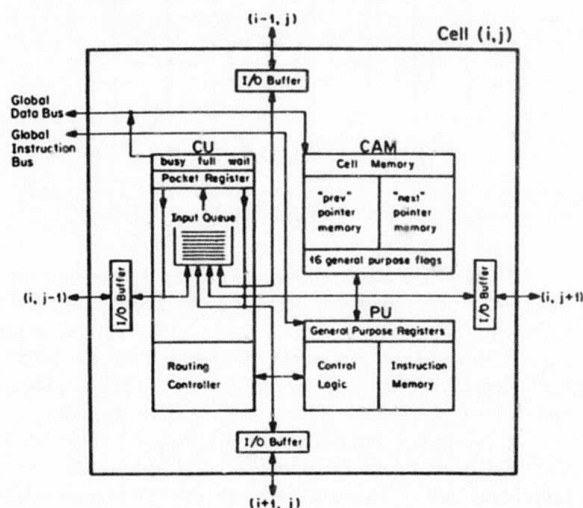


Figure 2-2: Architecture of SNAP Cell

2.2. Data Representation

A semantic network, here, is thought of as a colored graph. The nodes of the network represent objects or concepts and the arcs represent the relations between the objects. Usually, one cell is assigned to each node of the semantic network. The relations associated with that node are stored in the CAM. This realizes a doubly linked data structure allowing the operations to be carried out in either directions. An entry in the CAM consists of several fields. Complex matching and masking of individual fields can be achieved.

2.3. SNAP Instruction Set

The following instructions have been provided for processing symbolic data on SNAP. Their general format is $\langle \text{opCode} \rangle \langle \text{arg1} \rangle \langle \text{arg2} \rangle \langle \text{arg3} \rangle$. The first field $\langle \text{opCode} \rangle$ is the name of the function. The next two fields $\langle \text{arg1} \rangle$ and $\langle \text{arg2} \rangle$ can be flags, nodes, relations, or *don't cares*. If the argument is relation then it can have subfields and any of the subfields can be a *don't care*. We will use '%' to denote a *don't care* field. The third argument is generally a flag.

1. *Search*. The first argument is a node and the second one is a relation. The entries in the CAM corresponding to the relation ($\langle \text{arg1} \rangle \langle \text{arg2} \rangle$ %) are tagged and the cells which succeed in tagging a CAM entry will set their flag indicated by the third argument. For example, (search A R #1) will tag all CAM entries matching R in all the cells holding node A and set the flag 1 in those cells where a CAM entry was tagged.
2. *And, Or*. These are operations on flag or register arguments. They will perform the logical operations on the first two arguments and place the result in the third.
3. *Not*. will perform the logical negation of the first flag or register argument and place the result in the second.
4. *And-Cam (Or-Cam)*. The cell whose flag specified by $\langle \text{arg1} \rangle$ is set will AND (OR) together all the tagged entries in the CAM with the data specified by $\langle \text{arg2} \rangle$, and put the result in the register specified by $\langle \text{arg3} \rangle$.
5. *Compare*. The cells compare the registers specified by $\langle \text{arg1} \rangle$ and $\langle \text{arg2} \rangle$ for equality and set the flag specified by $\langle \text{arg3} \rangle$ if they are equal.
6. *F-search*. The cell whose flag given by $\langle \text{arg1} \rangle$ is set will send a message to the cells pointed by the tagged relations in its CAM to

set the flag $\langle \text{arg2} \rangle$ in them. The instruction *F-prop* is equivalent to *F-search* $\langle \text{arg1} \rangle$ $\langle \text{arg2} \rangle$ followed by *F-search* $\langle \text{arg2} \rangle$ $\langle \text{arg2} \rangle$ repeated until its failure. The instruction *F-prop* fails if the CAM has no tagged entries or the activity has cycled back to the originating cell(s).

7. *Collect,Collect-Relation*. *Collect* will retrieve the names of nodes from the cells whose flag given by $\langle \text{arg1} \rangle$ are set and bind the list to the variable $\langle \text{arg2} \rangle$. The instruction *Collect-Relation* works much the same as *Collect* except that it retrieves the tagged relations.
8. *Create*. This is the instruction for initialization of the array. It adds the relation $(\langle \text{arg1} \rangle \langle \text{arg2} \rangle \langle \text{arg3} \rangle)$ to the CAM of the cell holding $\langle \text{arg1} \rangle$. The node $\langle \text{arg1} \rangle$ or $\langle \text{arg2} \rangle$ will be created if it does not already exist.
9. *Delete*. The instruction will delete all the tagged entries in the CAM of the cell whose flag $\langle \text{arg1} \rangle$ is set.
10. *Send-Message*. The cell whose flag $\langle \text{arg1} \rangle$ is set will send the message given by $\langle \text{arg2} \rangle$ to the node specified by $\langle \text{arg3} \rangle$. If no destination is specified then the message is sent to the cells pointed by the tagged relations in the CAM. Notice that this instruction is more primitive than *F-prop* which uses this.
11. *Messages-Exist* is true if there are data packets in the message-queue of the cell.
12. *Reset-Tags* will reset all tags in the CAM of the cell whose flag $\langle \text{arg1} \rangle$ is set.

3. The Labeling Problem

The *labeling problem* is a problem of constraint satisfaction. Given a set of objects, a set of labels and a set of constraints, the problem is to assign a label to each object without violating any of the constraints. A labeling is called unambiguous if it assigns a single label to each object and it is consistent if it does not violate any constraints. An ambiguous labeling assigns a set of labels to each object. An ambiguous labeling is consistent as long as for every choice of a label for an object there exists at least one assigned label for every other object that does not violate any constraints. There may be none, one, or many unambiguous and consistent possible labelings. Some times one is interested in any unambiguous and consistent labeling and at other times in all possible solutions.

Depth-first search can be conducted to obtain an unambiguous and consistent labeling. However, with no guidelines for choosing the initial labelings, the search can be very inefficient. Sequential search can be speeded up by using the discrete relaxation technique [10]. In this technique, the label assignment that leads to an inconsistent labeling is removed, thus reducing the search space.

In this section we discuss the parallel processing of the labeling problem using the relaxation technique on SNAP. First, we present a model for the discrete relaxation operation and show how this model can be put in the form of a semantic network acceptable by SNAP. In SNAP Then, we show how the discrete relaxation can be processed on this architecture.

3.1. Problem Definition

What follows is a parallel model of Waltz filtering process as described by Rosenfeld *et. al.* [10]. Let $A = \{a_1, a_2, \dots, a_n\}$ be the set of objects to be labeled with the labels in the set $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_m\}$. Let the unary constraints on the labeling be such that $A_i \subseteq \Lambda$ be the set of possible labels for object a_i . Then for any pair of objects (a_i, a_j) , let $A_{ij} \subseteq A_i \times A_j$ denote the compatible labels in accordance with the binary constraints, i.e., $(\lambda, \lambda') \in A_{ij}$ means that it is possible to label the object a_i with λ and a_j with λ' without violating any constraints. General treatment of n-ary constraints can be found in [5]. In this paper we shall restrict ourselves to unary and binary constraints. A labeling $L = (L_1, \dots, L_n)$ of A is an assignment of a set of labels $L_i \subseteq \Lambda$ to each of $a_i \in A$. The labeling is *consistent* if we have

$$\forall i, j \quad \forall \lambda \in L_i \quad (\{\lambda\} \times L_j) \cap A_{ij} \neq \emptyset \quad (1)$$

A labeling $L = (L_1, \dots, L_n)$ is said to be included in another labeling $L' = (L'_1, \dots, L'_n)$ if $L_i \subseteq L'_i$ for all i . A labeling is the *greatest* if it is consistent and not included in any other consistent labeling. A labeling is *ambiguous* if it assigns more than one label to any object. It can be shown [10] that there always exist consistent labelings, in particular, the null and the greatest labelings.

The algorithm starts with an initial labeling $L^0 = (A_1, \dots, A_n)$. Let L^k be the labeling at k th iteration of the algorithm. Then the labeling L^{k+1} is obtained in the next iteration by discarding any label λ from each L_i^k such that $(\{\lambda\} \times L_j^k) \cap A_{ij} = \emptyset$ for some j , i.e., the label λ for object a_i violates the consistency condition (1). Due to finiteness of the problem domain the algorithm converges. It can be

shown [10] that the algorithm converges to the greatest consistent labeling.

The labeling problem has many applications. Haralick and Shapiro [6] have shown that the problems of subgraph isomorphism [12], scene labeling [1], shape matching [2] and many others are specific instances of the labeling problem. Rutkowski *et. al.* [11] have shown the applicability of the relaxation technique to scene segmentation.

3.2. Semantic Network Representation of the Labeling Problem

The labeling problem can be equivalently described by a graph model. Construct a colored directed graph $G=(V,L,E)$ where

1. V is the set of vertices such that $v_i \in V \Leftrightarrow a_i \in A$ for all i ,
2. L is the set of colors, $L = A \times A$, and
3. E is the set of edges, $E = \{ (v_i, v_j, l) \mid v_i, v_j \in V \text{ and } l \in A_{ij} \}$.

This graph represents a labeling $L=(L_1, \dots, L_n)$ where $L_i = \{ \lambda \mid (v_i, v_j, (\lambda, \lambda')) \in E \text{ for some } j \text{ and } \lambda, \lambda' \in A \}$. It is easily seen that the graph thus constructed represents a consistent labeling iff

$$\begin{aligned} \forall v \forall v' \neq v \quad (v, v', (\lambda, \lambda')) \in E \\ \Rightarrow \forall v'' \neq v \exists \lambda'' (v, v'', (\lambda, \lambda'')) \in E \end{aligned} \quad (2)$$

3.3. Discrete Relaxation on SNAP

In section 3.2 we showed how the discrete relaxation problem can be represented by a semantic network. Allocate one cell to each vertex in the semantic network. For this problem it suffices to store only the relations emanating from each node. The relation in our case is the tuple $(v (\lambda \lambda') v')$ where λ' is the label on the arc from node v to v' . To simplify checking for consistency condition we represented the vertices by bit vectors. The labels are encoded. The format of the CAM entries is $\langle \lambda \rangle \langle \lambda' \rangle \langle v' \rangle$.

In each iteration of the algorithm, the host computer scans through all the labels. Each cell tags all the relations that assign the label selected by the host to the object represented by the cell. If this assignment is not consistent then the tagged relations are deleted. The host then traces along those relations enabling the cells where the corresponding backward relations are stored. The backward relations are tagged and deleted. With the bit vector representation, the consistency check can be accomplished by logical OR and compare instructions. The program flow for a single iteration on SNAP is shown

below. The program will be terminated if the host finds that none of the #2 flags are set at the end of an iteration.

{Notes:

1. The instruction reset-flag is not a SNAP primitive. It can be written using AND and other logical operations.
2. Formation of the message in the send-message instruction can be done using other functions
3. V and N are registers. N holds the id of the node }

FOR ALL $\lambda \in A$

BEGIN

RESET-TAG #% { reset all flags and tags}
RESET-FLAG #%

SEARCH % (λ % %) #1
{ tag all relations
beginning with λ }

OR-CAM #1 N V { OR of all tagged
relations }

AND V 'vertex V { get vertex field of
the ORed result }

COMPARE V '111..1 N #2 { if the label
points to all
objects then set
flag 2 otherwise
the label is
inconsistent }

RESET-TAG #2 {if consistent then do
nothing to this cell }

NOT #2 #2

SEND-MESSAGE #2 (*search % (λ % %) N #2*)
{ send the message to tag
backward inconsistent
relations in the cells
pointed by inconsistent
relations }

DELETE #2 { Delete inconsistent
entries }

END.

Clearly, the running time for each iteration is $O(|A|)$. Due to possible bus contention during the send-message instruction, the total running time is much dependent on the structure of the semantic network and the node to cell allocation strategy. A good allocation policy is to preserve the locality of the nodes in the semantic network as much as possible. The details of applications of the discrete relaxation and the symbolic execution of the algorithm on SNAP can be found in the paper by Dixit and Moldovan [3].

4. Stereo and Dynamic Programming

This section illustrates how SNAP can be effectively used to solve problems requiring dynamic programming. The problem of stereo matching is used as a vehicle to do so. The definition problem and its solution are adopted from Ohta and Kanade [9]. We would not describe it here in great detail. The interested reader should consult the original report.

Stereo, whether edge-based or feature-based, has the basic problem of correspondence of the objects in one image to those in the other. Ohta and Kanade consider the problem of correspondence in edge-based stereo. They suggest a solution consisting of dynamic programming at two levels. On the inner level, the problem is solved on each scanline; on the outer level, the optimization is accomplished for the whole image. The goal is to find a path in 3D search space, which is a stack of 2D search planes. The inner level dynamic programming, called *intra-scanline* search, is to find optimal paths in 2D planes. A scanline from left image and another from right image form the axes of a 2D search plane. The outer level dynamic programming, called *inter-scanline* search, is to find an overall optimal matching surface (set of paths) under the constraint given by connected edges. The dynamic programming searches in both levels are conducted simultaneously.

4.1. Intra-scanline search on 2D plane

Let $m = (m, n)$ represent a node in the 2D search plane where m and n are the indices of the left and right image edges on the scanline. Let $D(m, k)$ be the minimal cost partial path from node k to node m . Let $d(m, k)$ be the cost of the primitive path from node k to node m . The cost of a primitive path is a function of intensity, color, orientation, and other directly measurable quantities of the edges involved. The minimal cost path from the origin $D(m, 0)$ can be defined recursively as:

$$D(m) = \min_{\{i\}} \{d(m, m-i) + D(m-i)\} \quad (3)$$

$$D(0) = 0$$

Here $m = (m, n), i = (i, j), 0 \leq i \leq m, 0 < j < n, i+j \neq 0$.

4.2. Inter-scanline search in 3D space

The 3D space is formed by the stack of the 2D planes for intra-scanline search. Since a 3D node is a set of 2D nodes, the cost of a 3D node is the sum of the costs of its 2D nodes obtained by intra-scanline search. To use dynamic programming, the 3D nodes must be ordered into decision stages. The ordering rule of Ohta and Kanade is: *When we examine the correspondence of the two connected edges, one in the right and one in the left image, the connected edges which are on the left of these connected edges in each image must already be processed.*

The connected edges are numbered from 0 to $U[V]$ in the left [right] image. Let u represent the 3D node (u, v) . The cost of optimal paths, $C(u)$, which come up to the 3D node u , is computed as follows:

$$C(u) = \min_{\{i\}} \sum_{t=s(u)}^{e(u)} \{D(l(u; t), l(u-i(t); t); t) + C(u-i(t); t)\} \quad (4)$$

$$C(0) = 0, \text{ i.e. } C(0; t) = 0 \text{ for all } t$$

$$\text{Here } u = (u, v), i(t) = (i(t), j(t)), 0 \leq i(t) \leq u, 0 \leq j(t) \leq v, i(t) + j(t) \neq 0.$$

Here $C(u, t)$ is the cost of the path on the scanline t in the optimal set and $C(u)$ is the sum of $C(u, t)$, t going from starting scanline $s(u)$ to ending scanline $e(u)$. $D(m, k; t)$ is the cost of the optimal 3D primitive path from node k to node m on the 2D plane for scanline t . The function $l(u; t)$ gives the index of a node belonging to the 3D node u on the 2D plane for scanline t .

4.3. Stereo on SNAP

Moldovan [7] describes how dynamic programming can be mapped onto a systolic array starting from a FORTRAN program. The same ideas can be used here to identify the data movements when the stereo problem is mapped onto SNAP. Although SNAP is more complex than a systolic array, it can be used in systolic mode by simply restricting one's programming style. Assumption here is that the host and the array controller are capable of issuing commands and controlling the external data movements in synchronous with the activity in the SNAP array. The solution we suggest here is not purely systolic, but a combination of pipelining, broadcasting, and associative memory operations.

Let us rewrite the equation (4) for computing the minimal cost path as FORTRAN program. In the following program two new variables A and B are introduced to expand the minimization operations into FORTRAN loops.

```

for u = 0 ... U
  for i = 0 ... u
    for t = s(u) ... e(u)
      for m = m0 ... m1
        for p = 0 ... m0-m1
          D(m, m0, t) = min(D(m-j, m0, t)
                           + d(m, m-j), D(m, m0, t))
        end
      end
      C(u, t) = C(u-i(t), t) + D(m1, m0, t)
      B(u, t) = B(u, t-1) + C(u, t)
    end
    A(u) = min {A(u), B(u, *)}
  end
end.
```

Note:

1. $m_0 = l(u-i(t); t)$
 $m_1 = l(u; t)$
2. Use of a vector as the index of a *for loop* simply denotes nested *for loops*, one loop for each component of the vector. In our case, the order of nesting is immaterial.

4.3.1. Data Representation and Cell Allocation

One row of SNAP is allocated to each scanline. Each cell in a row stores information about one edge in left(right) image, and the associative memory within each cell contains an entry for each possible matching edge in the right(left) image. An entry in the associative memory has the fields shown in Figure 4-1.

ID	INTER- DITY	ORIGIN- TATION	...	PREVIOUS POINT IN THE PATH	A	B	C	D
----	----------------	-------------------	-----	----------------------------------	---	---	---	---

(Note: A, B, C, and D are variable in the program)

Figure 4-1: CAM Entry for a Possible Matching Edge

The cells are assigned to the edges in right or left image such that the vertically (horizontally) connected edges are mapped onto same column (row) and consecutive rows (columns). This can be thought of as iconic representation of the image at edge level. If horizontally connected edges are present, broadcasting may be necessary to compute the summation over the *t-loop*, otherwise, simple pipelining would suffice.

4.3.2. Program Flow

The program on the host-SNAP combination is a parallelized and pipelined version of the sequential program. The different *for loops* are executed as follows:

- u-loop* : in parallel for all connected edges
- i-loop* : sequentially run by host
- t-loop* : summation is done sequentially by host
- m-loop* : in parallel for all edges on every scanline
- p-loop* : pipelined along rows

The host executes the *i-loop* sequentially selecting a point $i = (i, j)$. Each cell in the SNAP computes the limits for the *m-loop*. The pipelined execution of *p-loop* begins. The leftmost cell in each scanline starts sending the values of $D(*)$ of each CAM entry along with other information to the cell to its right. The other cells receive the values from the left cell and update their values of $D(*)$. The $D(*)$ values are updated for all entries in the CAM simultaneously by bitwise arithmetic operations. These operations are addition, multiplication, and taking

the minimum. Algorithms for such CAM computations are described by Foster [4]. After updating the $D(*)$ values, the packet received from left is pipelined to the right cell. When the cell originating the data packets for the pipeline sends its last packet, which is marked as such, the next cell becomes the originator. The *p-loop* ends when the rightmost cell becomes the originator. Now the summation over scanlines, the *t-loop*, is done serially by the host employing broadcasting and pipelining. A minimum is taken to compute the value of A and the process is iterated for another value of i .

In order to retrieve the optimal path, one must do a backward search: start from the final point (cell and CAM entry) in the path. Fetch the id of the previous point which produced this value. Go to that previous cell and so on until you reach the origin of the path.

4.3.3. Complexity

The maximum number of the total CAM entries in any row is not more than MN , the product of the maximum number of edges on any scanline in the two images. Similarly the time for *i-loop* is $O(UV)$, where U and V are the number of connected edges in the left and the right image respectively. The summation over scanlines takes the time proportional to maximum of the lengths of connected edges, which is $O(Tb)$, where T is the number of scanlines and b is the number of bits used for numbers. Thus the total time on SNAP is $O(UVMNTb)$. Compare this with the sequential execution time $O(U^2V^2M^2N^2T)$.

5. Conclusions

In this paper we presented an architecture intended for symbolic processing. We examined the applicability of SNAP to image understanding problems of scene labeling and stereo. The two problems are computationally intensive and employ different techniques for their solutions. Solution to Stereo uses *intra-* and *inter-scanline* dynamic programming technique where the computations are highly ordered. However, the discrete relaxation used for the labeling problem is a parallel technique. Since these are general techniques, any other problems that use them for their solutions could be solved on SNAP with the same speed up.

6. Acknowledgement

We are indebted to Ram Nevatia and Gerard Medioni for suggesting these two IU problems as SNAP applications.

References

- [1] Barrow, H. G. and Tanenbaum, J. M.
MYSYS: A system for Reasoning About Scenes.
AI 121, Stanford Research Institute, 1976.
- [2] Davis, L.
Shape Matching Using Relaxation Techniques.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-1(1):60-72, January, 1979.
- [3] Dixit, V. V. and Moldovan, D. I.
Discrete Relaxation on SNAP.
In *Proceedings of the First Conference on Artificial Intelligence Applications.* Denver, Co., December, 1984.
- [4] Foster, C.
Content Addressable Parallel Processors.
Van Nostard Reinhold, N.Y., 1976.
- [5] Haralick, R. M. and Kartus, J. S.
Arrangements, Homomorphisms, and Discrete Relation.
IEEE Transactions on Systems, Man, and Cybernetics SMC-8(8):600-612, August, 1978.
- [6] Haralick, R. M., and Shapiro, L. G.
The Consistent Labeling Problem: Part I.
IEEE Transaction on Pattern Analysis and Machine Intelligence PAMI-1(2):173-184, April, 1979.
- [7] Moldovan, D.I.
On the Design of Algorithms for VLSI Systolic Arrays.
Proceedings of the IEEE 71(1):113-120, January, 1983.
- [8] Moldovan, D.I. and Tung, Y-W.
SNAP: A VLSI Architecture for Artificial Intelligence Processing.
Technical Report PPP 84-3, USC, 1984.
- [9] Ohta, Y. and Kanade, T.
Stereo by Intra- and Inter-scanline Search Using Dynamic Programming.
Technical Report CMU-CS-83-162, CMU, 1983.
- [10] Rosenfeld, A., Hummel, R. A., and Zucker, S. W.
Scene Labeling by Relaxation Operations.
IEEE Transactions on Systems, Man, and Cybernetics SMC-6(6):420-433, June, 1976.
- [11] Rutkowski, W. S., Peleg, S., and Rosenfeld, A.
Shape Segmentation Using Relaxation.
IEEE Transaction on Pattern Analysis and Machine Intelligence PAMI-3(4):368-375, July, 1981.
- [12] Ullman, J. R.
An Algorithm for Subgraph Isomorphism.
JACM 23(1):31-42, January, 1976.

EXTENDED GAUSSIAN IMAGES

Berthold K. P. Horn

Artificial Intelligence Laboratory

Massachusetts Institute of Technology

Abstract

This is a primer on extended Gaussian Images. Extended Gaussian Images are useful for representing the shapes of surfaces. They can be computed easily from:

1. Needle maps obtained using photometric stereo, or
2. Depth maps generated by ranging devices or binocular stereo.

Importantly, they can also be determined simply from geometric models of the objects. Extended Gaussian images can be of use in at least two of the tasks facing a machine vision system:

1. Recognition, and
2. Determining the attitude in space of an object.

Here, the extended Gaussian image is defined and some of its properties discussed. An elaboration for non-convex objects is presented and several examples are shown.

1. Introduction

In order to recognize an object and to determine its attitude in space, it is necessary to have a way of representing the shape of its surface. Giving the distance to the surface along parallel rays on a regularly spaced grid provides one way of doing this. This simple representation is called a depth map. A range finder produces surface descriptions in this form, as does automated binocular stereo [1]. Unfortunately, depth maps do not transform in a simple way when the object rotates (For one thing, interpolation must be used to get a new depth map on a regularly spaced grid).

Alternatively, surface orientation might be given for points on the surface on some regular sampling grid. This grid may conveniently correspond to the picture cells in an image. Such a simple representation is called a needle map (Figure 1) [2]. Photometric stereo is a method for recovering surface orientation using multiple images taken with different lighting conditions [3-8]. It produces surface descriptions in this form. A needle map also is not directly

helpful when it comes to comparing surfaces of objects that may be rotated relative to one another (Both depth maps and needle maps depend on the *position* of the object as well as its attitude).

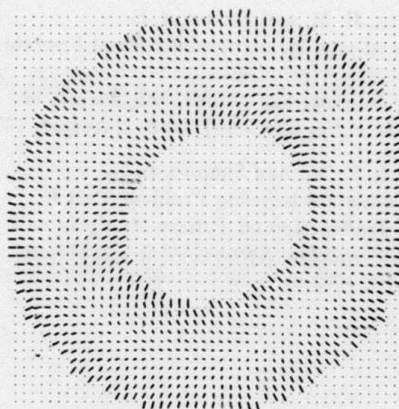


Figure 1. A needle map shows unit surface normals at points on the surface on a regular grid. Normals which point towards the viewer will be seen as dots, while tilted surface patches give rise to normals which are shown as lines pointing in the direction of steepest descent.

The extended Gaussian image, on the other hand, does make it easy to deal with the varying attitude of an object in space as we shall see [2, 9-13]. For one thing, it is insensitive to the position of the object. Some information appears to be discarded in the formation of the extended Gaussian image. Curiously, in the case of convex objects, the representation is nevertheless unique. That is, no two convex objects have the same extended Gaussian image.

This representation of the shape of the surface of an object allows one to match information obtained from image or range sensors with that contained in computer models of the objects and has proven most

useful in work on automatic bin picking [14-16]. A recent report describes a system that picks one object out of a jumbled pile of similar objects using this approach [17]. We start our discussion with objects having planar faces. Later we consider smoothly curved objects. Methods for computing discrete approximations of extended Gaussian images, called orientation histograms, are presented too. Orientation histograms can be computed from experimental data or mathematical descriptions of the objects. Sections marked with an asterisk may be omitted on first reading or if your interest in the mathematical details is limited.

2. DiscreteCase: Convex Polyhedra

Minkowski showed in 1897 that a convex polyhedron is fully specified (up to translation) by the area and orientation of its faces [18-20]. We can represent area and orientation of the faces conveniently by point masses on a sphere. Imagine moving the unit surface normal of each face so that its tail is at the center of a unit sphere. The head of the unit normal then lies on the surface of the unit sphere. This sphere is called the Gaussian sphere and each point on it corresponds to a particular surface orientation. The extended Gaussian image of the polyhedron is obtained by placing a mass at each point equal to the surface area of the corresponding face (Figure 2).

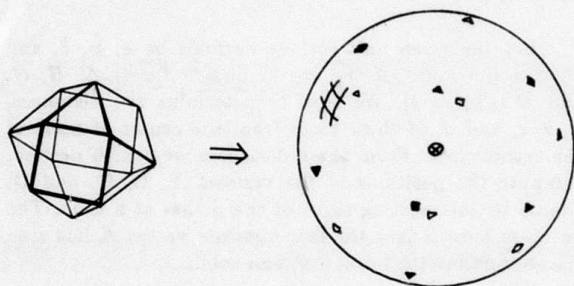


Figure 2. The extended Gaussian image of a polyhedron can be thought of as a collection of point masses on the Gaussian sphere. Each mass is proportional to the area of the corresponding face. Point masses on the visible hemisphere are shown as solid marks, the others as open marks. The center of mass (shown as the symbol \otimes) must be at the center of the sphere if the polyhedron is a closed.

It seems at first, as if some information is lost in this mapping, since the *position* of the surface normals is discarded. Viewed another way, no note is made of the shape of the faces or their adjacency relationships. It can nevertheless be shown that (up to translation) the extended Gaussian image uniquely defines a convex polyhedron [9]. An iterative algorithm has recently been invented for recovering a convex polyhedron from its extended Gaussian image [21].

2.1. Properties of the Extended Gaussian Image

The extended Gaussian image is not affected by translation of the object. Rotation of the object induces an equal rotation of the extended Gaussian image, since the unit surface normals rotate with the object.

Mass distributions which lie entirely within one hemisphere, that is, are zero in the complementary hemisphere, do not correspond to closed objects. As we shall see, the center of mass of an extended Gaussian image has to lie at the origin. This is clearly not possible if a whole hemisphere is empty. Also, a mass distribution which is non-zero only on a great circle of the sphere corresponds to the limit of a sequence of cylindrical objects of increasing length and decreasing diameter (Figure 3). We will exclude such pathological cases and confine our attention to closed, bounded objects [9, 20].

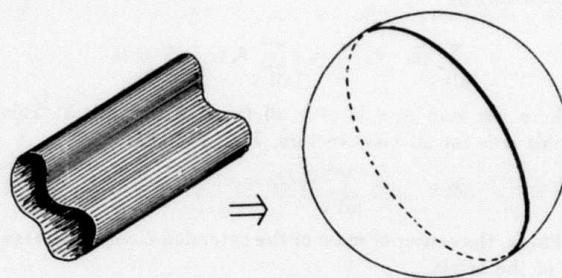


Figure 3. A mass distribution confined to a great circle corresponds to the limit of a sequence of cylindrical objects of increasing length and decreasing diameter. Such pathological mass distributions can be avoided if we confine our attention to bounded objects.

Some properties of the extended Gaussian image are important: Firstly, the total mass of the extended Gaussian image is obviously just equal to the total surface area of the polyhedron. If the polyhedron is closed, it will have the same projected area when viewed from any pair of opposite directions. This allows us to compute the location of the center of mass of the extended Gaussian image.

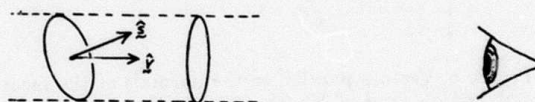


Figure 4. A surface element appears smaller because of foreshortening. The apparent area is the true area times the cosine of the angle between the surface normal and the vector pointing towards the viewer.

Imagine viewing a convex polyhedron from a great distance. Let the direction from the object towards the

viewer be given by the unit vector \hat{v} . A face, with unit normal \hat{s}_i , will be visible only if $\hat{s}_i \cdot \hat{v} \geq 0$. Suppose that the surface area of this face is O_i . Due to foreshortening it will appear only as large as would a face of area

$$(\hat{s}_i \cdot \hat{v}) O_i,$$

normal to \hat{v} (Figure 4). The total apparent area of the visible surface is

$$A(\hat{v}) = \sum_{\{\hat{s}_i \cdot \hat{v} \geq 0\}} (\hat{s}_i \cdot \hat{v}) O_i,$$

when viewed from the direction \hat{v} . The total apparent area of the visible surface when viewed from the opposite direction is

$$A(-\hat{v}) = \sum_{\{\hat{s}_i \cdot \hat{v} \leq 0\}} (\hat{s}_i \cdot \hat{v}) O_i,$$

This should be the same, that is, $A(\hat{v}) = A(-\hat{v})$. Consequently,

$$\sum_{\text{all } i} (\hat{s}_i \cdot \hat{v}) O_i = \left[\sum_{\text{all } i} \hat{s}_i O_i \right] \cdot \hat{v} = 0,$$

where the sum now is over all faces of the object. This holds true for all view vectors, \hat{v} , so we must have

$$\sum_{\text{all } i} \hat{s}_i O_i = 0.$$

That is, the center of mass of the extended Gaussian image is at the origin.

An equivalent representation, called a spike model, is a collection of vectors each of which is parallel to one of the surface normals and of length equal to the area of the corresponding face. The result regarding the center of mass is equivalent to the statement that these vectors must form a closed chain when placed end to end (Figure 5).

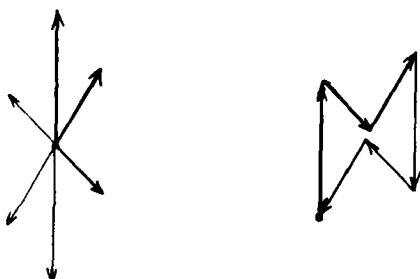


Figure 5. Vectors parallel to the normals of the faces of a polyhedron, and of length equal to the areas of the corresponding faces, form a closed chain when placed end to end.

2.2. Reconstruction of a Tetrahedron (*)

Faces that share a common edge are said to be adjacent. The masses on the Gaussian sphere corresponding to two adjacent faces need not be each others closest neighbors. Recovering a polyhedron from its extended Gaussian image

is not easy in the general case because it is hard to determine which faces are adjacent [21]. Finding the actual offsets of each of the faces from the center of mass of the polyhedron is not as hard.

The structure of a tetrahedron, however, is very simple: Every face is adjacent to the other three. The shape of the tetrahedron is completely determined by the surface normals of the four faces, only the size of the tetrahedron remaining to be determined. In other words: There is only one degree of freedom left. Another way to look at it is to note that the four faces must have areas that place the center of mass of the extended Gaussian image at the origin, as we have just seen. This condition places three constraints on the four parameters.

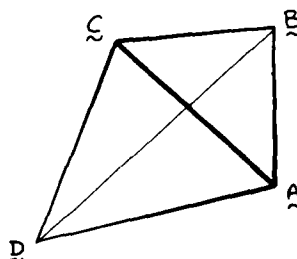


Figure 6. A tetrahedron with vertices A, B, C, and D. We are to find the distances of the faces from the center of mass, given the areas and surface normals of the faces.

Let the given unit surface normals be \hat{a} , \hat{b} , \hat{c} , and \hat{d} , and the areas of the corresponding faces, A, B, C, and D (Figure 6). We have to determine the distances, a , b , c , and d , of these faces from the center of mass of the tetrahedron. From these distances we can, if desired, compute the positions of the vertices A, B, C, and D, simply by intersecting three of the planes at a time. The notation here is that the face opposite vertex A has area A and unit surface normal \hat{a} , and so on.

The perpendicular distance of the center of area of a triangle from one of the sides is one third the perpendicular distance of the vertex opposite that side. Similarly, in a tetrahedron, the distance from the center of mass to a particular face is equal to one quarter of the distance of the vertex opposite that face. We start by finding a formula for the distance of the face with area D, say, from the opposite vertex d. The desired distance, d , will be just a quarter of the result obtained in this fashion. The remaining three distances, a , b , and c , can then be computed using formulae obtained by cyclical permutation of the variables.

The position of the reconstructed tetrahedron is arbitrary, since the extended Gaussian image is insensitive to translation. To make the result unique, we might place the center of mass at the origin. To reduce the size of the expressions to be manipulated here, however, it is convenient to move the tetrahedron so that one vertex, D, say, is at the origin. The distances of the faces from the center of mass are obviously not affected by this.

Suppose for now that we know the locations of the vertices A , B , and C relative to D . We can then compute the directions of the six edges of the tetrahedron, by taking all of the distinct pairwise differences of the four vertex positions. Four surface normals can then be found by taking cross-products of these edge-direction vectors. We actually need only four of the edge vectors forming a closed circuit to do this. The results can then be normalized to obtain unit surface normals,

$$\hat{a} = -\frac{\mathbf{B} \times \mathbf{C}}{|\mathbf{B} \times \mathbf{C}|}, \quad \hat{b} = -\frac{\mathbf{C} \times \mathbf{A}}{|\mathbf{C} \times \mathbf{A}|}, \quad \hat{c} = -\frac{\mathbf{A} \times \mathbf{B}}{|\mathbf{A} \times \mathbf{B}|},$$

and,

$$\hat{d} = \frac{(\mathbf{A} - \mathbf{C}) \times (\mathbf{B} - \mathbf{A})}{|(\mathbf{A} - \mathbf{C}) \times (\mathbf{B} - \mathbf{A})|} = \frac{\mathbf{A} \times \mathbf{B} + \mathbf{B} \times \mathbf{C} + \mathbf{C} \times \mathbf{A}}{|\mathbf{A} \times \mathbf{B} + \mathbf{B} \times \mathbf{C} + \mathbf{C} \times \mathbf{A}|}.$$

Now the perpendicular distance of the plane with area D from the origin can be found by taking the dot-product of any of the three vertices, A , B , and C with the unit normal \hat{d} . Thus,

$$4d = \hat{d} \cdot \mathbf{A} = \hat{d} \cdot \mathbf{B} = \hat{d} \cdot \mathbf{C} = \frac{[\mathbf{A} \mathbf{B} \mathbf{C}]}{|\mathbf{A} \times \mathbf{B} + \mathbf{B} \times \mathbf{C} + \mathbf{C} \times \mathbf{A}|}.$$

The area of the facet opposite the origin is also easy to compute,

$$D = \frac{1}{2}|(\mathbf{A} - \mathbf{C}) \times (\mathbf{B} - \mathbf{A})| = \frac{1}{2}|\mathbf{A} \times \mathbf{B} + \mathbf{B} \times \mathbf{C} + \mathbf{C} \times \mathbf{A}|.$$

Our task is to express the offset d in terms of the area D and the given unit surface normals. The two formulae above do not allow us to do that directly, because we do not know what the value of $[\mathbf{A} \mathbf{B} \mathbf{C}]$ is. This quantity, by the way, is six times the volume, V , of the tetrahedron, or,

$$V = \frac{1}{3}(4d)D = \frac{1}{6}[\mathbf{A} \mathbf{B} \mathbf{C}],$$

We proceed by considering the four distinct triple products of the four unit surface normals. First of all,

$$[\hat{a} \hat{b} \hat{c}] = -\frac{[\mathbf{A} \mathbf{B} \mathbf{C}]^2}{|\mathbf{A} \times \mathbf{B}| |\mathbf{B} \times \mathbf{C}| |\mathbf{C} \times \mathbf{A}|},$$

since $[(\mathbf{x} \times \mathbf{y})(\mathbf{y} \times \mathbf{z})(\mathbf{z} \times \mathbf{x})] = [\mathbf{x} \mathbf{y} \mathbf{z}]^2$. Then, by similar reasoning,

$$[\hat{a} \hat{b} \hat{d}] = \frac{[\mathbf{A} \mathbf{B} \mathbf{C}]^2}{|\mathbf{B} \times \mathbf{C}| |\mathbf{C} \times \mathbf{A}| |\mathbf{A} \times \mathbf{B} + \mathbf{B} \times \mathbf{C} + \mathbf{C} \times \mathbf{A}|},$$

since $[\mathbf{x} \mathbf{y} (\mathbf{x} + \mathbf{y} + \mathbf{z})] = [\mathbf{x} \mathbf{y} \mathbf{z}]$. Formulae for $[\hat{b} \hat{c} \hat{d}]$ and $[\hat{c} \hat{a} \hat{d}]$ can be found by cyclical permutation of the variables.

Multiplying the three formulae found this way together we get,

$$[\hat{a} \hat{b} \hat{d}][\hat{b} \hat{c} \hat{d}][\hat{c} \hat{a} \hat{d}] = \frac{[\mathbf{A} \mathbf{B} \mathbf{C}]^6}{(\mathbf{A} \times \mathbf{B})^2 (\mathbf{B} \times \mathbf{C})^2 (\mathbf{C} \times \mathbf{A})^2 |\mathbf{A} \times \mathbf{B} + \mathbf{B} \times \mathbf{C} + \mathbf{C} \times \mathbf{A}|^3},$$

and so

$$\frac{[\hat{a} \hat{b} \hat{d}][\hat{b} \hat{c} \hat{d}][\hat{c} \hat{a} \hat{d}]}{[\hat{a} \hat{b} \hat{c}]^2} = \frac{[\mathbf{A} \mathbf{B} \mathbf{C}]^2}{|\mathbf{A} \times \mathbf{B} + \mathbf{B} \times \mathbf{C} + \mathbf{C} \times \mathbf{A}|} = (4d)^2(2D).$$

So that finally,

$$4d = \frac{\sqrt{(2D)[\hat{a} \hat{b} \hat{d}][\hat{b} \hat{c} \hat{d}][\hat{c} \hat{a} \hat{d}]}}{-[\hat{a} \hat{b} \hat{c}]}.$$

The other distances, a , b , and c , can be computed using similar formulae obtained by cyclical permutation of the variables.

3. Continuous Case: Smoothly Curved Objects

The ideas presented in the previous chapter can be extended to apply to smoothly curved surfaces.

3.1. Gaussian Image

One can associate a point on the Gaussian sphere with a given point on a surface by finding the point on the sphere which has the same surface normal (Figure 7) [20, 22, 25]. Thus it is possible to map information associated with points on the surface onto points on the Gaussian sphere. In the case of a convex object with positive Gaussian curvature everywhere, no two points have the same surface normal. The mapping from the object to the Gaussian sphere in this case is invertible: Corresponding to each point on the Gaussian sphere there is a unique point on the surface (If the convex surface has patches with zero Gaussian curvature, curves or even areas on it may correspond to a single point on the Gaussian sphere).

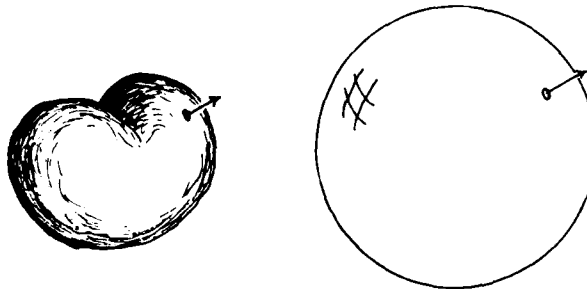


Figure 7. The Gaussian image of an object is obtained by associating with each point on its surface the point on the Gaussian sphere which has the same surface orientation. The mapping is invertible if the object has positive Gaussian curvature everywhere.

One useful property of the Gaussian image is that it rotates with the object. Consider two parallel surface normals, one on the object and the other on the Gaussian sphere. The two normals will remain parallel if the object and the Gaussian sphere are rotated in the same fashion. A rotation of the object thus corresponds to an equal rotation of the Gaussian sphere.

3.2. Gaussian Curvature

Consider a small patch δO on the object. Each point in this patch corresponds to a particular point on the Gaussian sphere. The patch δO on the object maps into a patch, δS say, on the Gaussian sphere (Figure 8). If the surface is strongly curved, the normals of points in the patch will point into a wide fan of directions. The corresponding points on the Gaussian sphere will be spread out. Conversely, if the surface is planar, the surface normals are parallel and map into a single point.

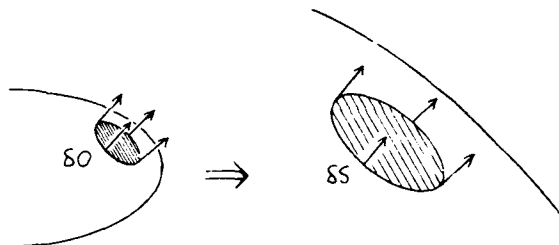


Figure 8. A patch on the object maps into a patch on the Gaussian sphere. The Gaussian curvature is the limit of the ratio of the area of the patch on the Gaussian sphere to the area of the patch on the object as these become smaller and smaller.

These considerations suggest a suitable definition of curvature. The Gaussian curvature is defined to be equal to the limit of the ratio of the two areas as they tend to zero. That is,

$$K = \lim_{\delta O \rightarrow 0} \frac{\delta S}{\delta O} = \frac{dS}{dO}.$$

From this differential relationship we can obtain two useful integrals. Consider first integrating K over a finite patch O on the object:

$$\int_O \int K dO = \int_S \int dS = S,$$

where S is the area of the corresponding patch on the Gaussian sphere. The expression on the left is called the integral curvature. This relationship allows one to deal with surfaces which have discontinuities in surface normal.

Now consider instead integrating $1/K$ over a patch S on the Gaussian sphere:

$$\int_S \int 1/K dS = \int_O \int dO = O,$$

where O is the area of the corresponding patch on the object. This relationship suggests the use of the *inverse* of the Gaussian curvature in the definition of the extended Gaussian image of a smoothly curved object, as we shall see. It also shows, by the way, that the integral of $1/K$ over the whole Gaussian sphere equals the total area of the object.

3.3. Alternate Definition of Gaussian Curvature

Consider a plane which includes the surface normal at some point on a smooth surface. The surface cuts this plane along a curve called a normal section (Figure 9) [19, 22-25]. Let the curvature of the normal section be denoted by κ_N . Consider the one-parameter family of planes containing the surface normal. Suppose that θ is the angle between a particular plane and a given reference plane. Then κ_N varies with θ in a periodic fashion. In fact, if we measure θ from the plane that gives maximum curvature, then it can be shown that

$$\kappa_N(\theta) = \kappa_1 \cos^2 \theta + \kappa_2 \sin^2 \theta,$$

where κ_1 is the maximum and κ_2 is the minimum curvature. These two values of κ are called the principal curvatures. The corresponding planes are called the principal planes. The two principal planes are orthogonal, provided that the principal curvatures are distinct (Figure 9).

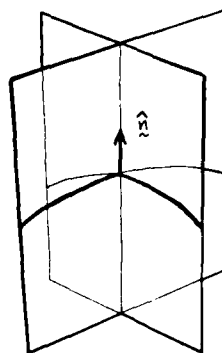


Figure 9. Normal sections of the surface are made with planes which include the surface normal. The planes corresponding to the largest and smallest value of curvature are referred to as the principal planes. The Gaussian curvature is equal to the product of the largest and smallest values of curvature.

It turns out that

$$K = \kappa_1 \kappa_2$$

is equal to the Gaussian curvature introduced earlier. This is clearly zero for a plane. It is equal to $1/R^2$ for a spherical surface of radius R , since the curvature of any normal section is $1/R$.

A ruled surface is one which can be generated by sweeping a line through space. A hyperboloid provides one example of such a surface. Developable surfaces are special cases of ruled surfaces [22, 23, 25]. Cylindrical and conical surfaces are examples of developable surfaces (Figure 10). On a developable surface at least one of the two principal curvatures is zero at all points. Consequently the Gaussian curvature is zero everywhere too.

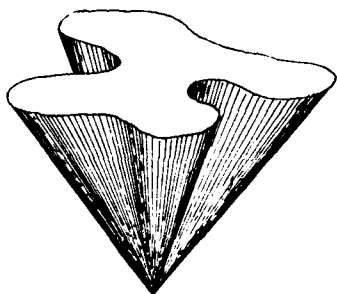


Figure 10. A conical surface is an example of a developable surface. On it the Gaussian curvature is everywhere zero, because (at least) one of the principal curvatures is zero.

3.4. The Extended Gaussian Image

We can define a mapping which associates the inverse of the Gaussian curvature at a point on the surface of the object with the corresponding point on the Gaussian sphere. Let u , and v be parameters used to identify points on the original surface. Similarly, let ξ and η be parameters used to identify points on the Gaussian sphere (These could be longitude and latitude, for example). Then we define the extended Gaussian image as

$$G(\xi, \eta) = \frac{1}{K(u, v)},$$

where (ξ, η) is the point on the Gaussian sphere which has the same normal as the point (u, v) on the original surface. It can be shown that this mapping is unique (up to translation) for convex objects. That is, there is only one convex object corresponding to a particular extended Gaussian image [9, 19, 26]. The proof is unfortunately non-constructive and no direct method for recovering the object is known.

3.5. Properties of the Extended Gaussian Image

The center of mass of the extended Gaussian image of a smoothly curved object is at the origin. We show this in a way similar to that used earlier for extended Gaussian images of polyhedral objects. Consider viewing a convex object from a great distance. Let the direction from the object towards the viewer be given by the unit vector \hat{v} . A surface patch with unit normal \hat{s} will be visible only if $\hat{s} \cdot \hat{v} \geq 0$. Suppose its surface area is δO (Figure 4). Due to foreshortening it will appear only as large as would a patch of area

$$(\hat{s} \cdot \hat{v}) \delta O,$$

normal to \hat{v} . Let $H(\hat{v})$ be the unit hemisphere for which $\hat{s} \cdot \hat{v} \geq 0$. Then the apparent area of the visible surface is

$$A(\hat{v}) = \int_{H(\hat{v})} \int G(\hat{s}) (\hat{s} \cdot \hat{v}) dS,$$

when viewed from the direction \hat{v} . The apparent area of the visible surface when viewed from the opposite direction is

$$A(-\hat{v}) = \int_{H(-\hat{v})} \int G(\hat{s}) (\hat{s} \cdot -\hat{v}) dS.$$

This should be the same, that is, $A(\hat{v}) = A(-\hat{v})$. Consequently,

$$\int_S \int G(\hat{s}) (\hat{s} \cdot \hat{v}) dS = \left(\int_S \int G(\hat{s}) \hat{s} dS \right) \cdot \hat{v} = 0,$$

where the integral now is over the whole sphere S . This holds true for all view vectors, \hat{v} , so we must have

$$\int_S \int G(\hat{s}) \hat{s} dS = 0.$$

That is, the center of mass of the extended Gaussian image is at the origin (This, by the way, is not a very helpful constraint in practice, since one usually only sees one side of the object).

Another property of the extended Gaussian image is also easily demonstrated. The total mass of the extended Gaussian image equals the total surface area of the object. If one wishes to deal with objects of the same shape but differing size one may normalize the extended Gaussian image by dividing by the total mass.

We can think of the extended Gaussian image in terms of mass density on the Gaussian sphere. It is possible then to deal in a consistent way with places on the surface where the Gaussian curvature is zero, using the integral of $1/K$ shown earlier. A planar region, for example, corresponds to a point mass. This in turn corresponds to an impulse function on the Gaussian sphere with magnitude proportional to the area of the planar region.

A mass distribution has inertia about an axis passing through its center of mass that depends on the direction of the axis. This inertia takes on three stationary values, for three particular orthogonal directions, called the principal axes of the object. It is tempting to imagine that one can find the attitude of an object by lining up the principal axes of inertia of the observed extended Gaussian image and the one computed from the geometric model [9]. This would be rather straightforward, requiring only the calculation of the eigenvectors of a three by three inertia matrix. In practice, one typically has information only about the visible hemisphere and thus cannot compute the required first and second moments over the whole sphere.

3.6. Objects that are not Convex

Three things happen when the surface is non-convex:

1. The Gaussian curvature for some points will be negative.
2. More than one point on the object will contribute to a given point on the Gaussian sphere.
3. Parts of the object may be obscured by other parts.

We chose to extend the definition of the extended Gaussian image in this case to be the sum of the *absolute* values of the inverses of the Gaussian curvature at all points having the same surface orientation,

$$G(\xi, \eta) = \sum_i \frac{1}{|K(u_i, v_i)|}.$$

This definition is motivated by the method used to compute the extended Gaussian image in the discrete case, as we will see later.

The above extension makes sense if there are a finite, or at most a countable, number of points on the surface with the same orientation. At times, however, all points on a curve or even an area on the surface have parallel surface normals. In this case we may use,

$$G(\hat{n}) = \int_O \int \delta(\hat{n} - \hat{s}) dO,$$

where \hat{n} is a unit vector on the Gaussian sphere, while \hat{s} is a unit vector on the surface of the object. The integration is over the whole surface of the object O and δ is the unit impulse function defined on a sphere.

We can be more specific, if we let $r(u, v)$ be a vector giving the point on the surface corresponding to the parameters u and v , then

$$G(\xi, \eta) \cos \eta = \int \int \delta(\xi - \theta(u, v), \eta - \phi(u, v)) |r_u \times r_v| du dv,$$

where $\theta(u, v)$ and $\phi(u, v)$ are the latitude and longitude of the point on the Gaussian sphere which has the same orientation as the surface does at the point (u, v) . A planar region of area A will thus contribute an impulse of weight A to the extended Gaussian image, while a cylindrical region will give rise to an impulse wall along a great circle at right angles to the axis of the cylinder. The integral of the impulse wall will be equal to the area of the cylindrical region.

Usually we think of the extended Gaussian image as a fixed entity associated with an object. In the case of non-convex objects we might want to alter the definition to include only those parts of the surface visible from a particular direction. This would make the (modified) Gaussian image dependent on the view-point. We avoid this potential complication here.

3.7. Examples of Extended Gaussian Images (*)

The extended Gaussian image of a sphere of radius R is

$$G(\xi, \eta) = R^2,$$

as discussed already.

Perhaps slightly more interesting is the case of an ellipsoid with semi-axes a , b and c lined up with the coordinate axes (Figure 11). An equation for its surface can be written

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 = 1.$$

More useful for our purposes here is a parametric form

$$\begin{aligned} x &= a \cos \theta \cos \phi, \\ y &= b \sin \theta \cos \phi, \\ z &= c \sin \phi. \end{aligned}$$

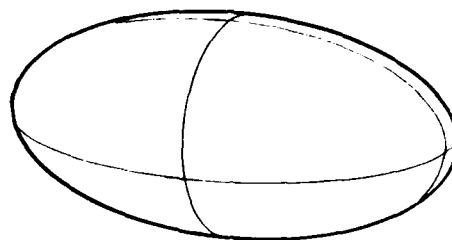


Figure 11. Ellipsoid with contours obtained by cutting the surface with three orthogonal planes passing through pairs of points where the Gaussian curvature has stationary values.

A normal at the point

$$r = (a \cos \theta \cos \phi, b \sin \theta \cos \phi, c \sin \phi)^T$$

on the surface is given by

$$n = (bc \cos \theta \cos \phi, ca \sin \theta \cos \phi, ab \sin \phi)^T,$$

as will be shown later. The Gaussian curvature turns out to be equal to

$$\begin{aligned} K &= \left[\frac{abc}{(bc \cos \theta \cos \phi)^2 + (ca \sin \theta \cos \phi)^2 + (ab \sin \phi)^2} \right]^2 \\ &= \left[\frac{abc}{n^2} \right]^2, \end{aligned}$$

where $n^2 = n \cdot n$.

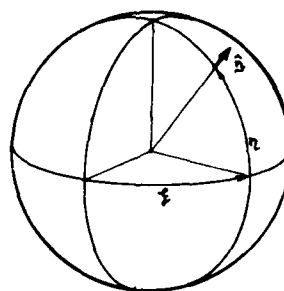


Figure 12. Latitude and longitude can be used to identify points on the Gaussian sphere. Each point on the Gaussian sphere corresponds to a unique surface orientation.

If we let ξ be the longitude and η be the latitude on the Gaussian sphere, then a unit normal at the point (ξ, η) on the sphere is given by (Figure 12)

$$\hat{n} = (\cos \xi \cos \eta, \sin \xi \cos \eta, \sin \eta)^T.$$

Now $n = n\hat{n}$. Identifying terms in the two expressions for surface normals at corresponding points on the ellipsoid and the Gaussian sphere we get,

$$\begin{aligned} bc \cos \theta \cos \phi &= n \cos \xi \cos \eta, \\ ca \sin \theta \cos \phi &= n \sin \xi \cos \eta, \\ ab \sin \phi &= n \sin \eta, \end{aligned}$$

so that

$$n^2[(a \cos \xi \cos \eta)^2 + (b \cos \xi \sin \eta)^2 + (c \sin \eta)^2] = (abc)^2,$$

and finally, substituting for n^2 in the equation for K we get

$$G(\xi, \eta) = \frac{1}{K} = \left[\frac{abc}{(a \cos \xi \cos \eta)^2 + (b \sin \xi \cos \eta)^2 + (c \sin \eta)^2} \right]^2.$$

The extended Gaussian image, in this case, varies smoothly and has the stationary values

$$\left(\frac{bc}{a}\right)^2, \left(\frac{ca}{b}\right)^2, \text{ and } \left(\frac{ab}{c}\right)^2,$$

at the points where r is equal to $(\pm 1, 0, 0)^T$, $(0, \pm 1, 0)^T$, and $(0, 0, \pm 1)^T$, respectively. These results can be easily checked by sectioning the ellipsoid using the xy , yz and zx -planes. The Gaussian curvature, in this case, equals the product of the curvatures of the resulting ellipses. One then uses the fact that the maximum and minimum curvatures of an ellipse with semi-axes a and b are a/b^2 and b/a^2 .

Later we will derive the extended Gaussian image of a torus, an object that is not convex.

4. Discrete Approximation: Needle Maps

Consider the surface broken up into small patches of equal area. Let there be ρ patches per unit area. Erect a surface normal on each patch. Consider the polyhedral object formed by the intersection of the tangent planes perpendicular to these surface normals. It approximates the original surface. The smaller the patches, the better the approximation.

The extended Gaussian image of the original (smoothly curved) convex object is approximated by impulses corresponding to the small patches. The magnitude of each impulse is about $1/\rho$, corresponding to the area of the patch it rests on (Figure 13). Strongly curved areas will distribute their impulses over a large region on the Gaussian sphere, while areas which are nearly planar will have them concentrated in a small region. In fact, the number of impulses per unit area on the Gaussian sphere approaches ρ times the absolute value of the Gaussian curvature as we make ρ larger and larger. This can be shown using the integral of $1/K$ given earlier.

The tessellation of the surface can be based on an arbitrary division into triangular patches as long as the magnitude of each impulse on the Gaussian sphere is made proportional to the area of the corresponding patch on the surface. Alternatively, one can divide the surface up according to the division of the image into picture cells. In this case one has to take into account that the area occupied in the image by a given patch is affected by foreshortening. The actual surface area is proportional to

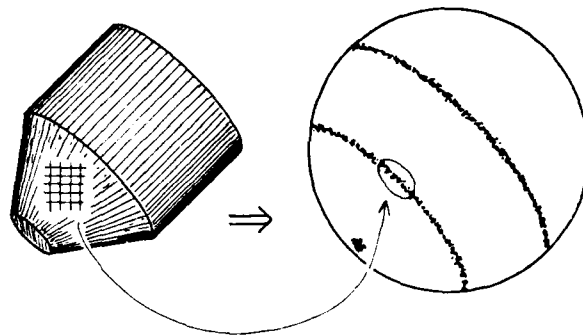


Figure 13. Mapping of discrete patches on an object onto the Gaussian sphere. The patches in this case correspond to a regular tessellation of the image plane. Since the patches lie on a conical surface they contribute to the extended Gaussian image along a small circle.

$1/(s_i \cdot \hat{v})$, where s_i is the normal of the patch, while \hat{v} is the vector pointing towards the viewer (Figure 4).

Measurements of surface orientation from images will not be perfect, since they are affected by the noise in brightness measurements. Similarly, surface orientations obtained from range data will be somewhat inaccurate. Consequently the impulses on the Gaussian sphere will be displaced a little from their true positions. The expected density on the Gaussian sphere will nevertheless tend to be equal to the inverse of the Gaussian curvature. One cannot, however, expect the impulses corresponding to a planar surface to be coincident. Instead, they will tend to form a small cluster. To be precise, the effect of noise is to smear out the information on the sphere. The extended Gaussian image is convolved with a smoothing function of width proportional to the magnitude of the noise.

4.1. Using Object Models

Extended Gaussian images also have to be computed for surfaces of prototypical object models. In this case it is best to find a convenient way to parameterize the surface and break it up into many small patches. Suppose the surface is given in terms of two parameters u and v as $r(u, v)$. Then at the point (u, v) we see that r_u and r_v are two tangents (Figure 14). The cross product of these tangents is normal to the surface. The unit normal

$$\hat{n} = \frac{r_u \times r_v}{|r_u \times r_v|}$$

allows us to determine to which point on the Gaussian sphere this patch corresponds. Suppose that we divided the range of u into segments of size δu and the range of v into segments of size δv . Then the area of the patch,

$$\delta A = |r_u \times r_v| \delta u \delta v,$$

can be used to determine what contribution it makes to the corresponding place on the Gaussian sphere. Note that we do not have to explicitly compute the Gaussian curvature or take second partial derivatives.

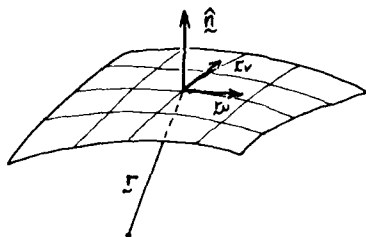


Figure 14. A surface normal can be computed by taking the cross-product of two tangent vectors. The tangent vectors can be obtained by differentiation of the parametric form of the equation of the surface.

5. Tessellation of the Gaussian sphere: Orientation Histograms

It is useful to divide the sphere up into cells in order to represent the information on the Gaussian sphere in a computer. Ideally the cells should satisfy the following criteria:

1. All cells should have the same area.
2. All cells should have the same shape.
3. The cells should have regular shapes that are compact.
4. The division should be fine enough to provide good angular resolution.
5. For some rotations the cells should be brought into coincidence with themselves.

Cells which are compact combine information only from surface patches which have nearly the same orientation. Elongated cells of the same area combine information from surface patches which have more widely differing orientations. The area of a regular polygon with n sides inscribed in a circle of radius r is

$$\pi r^2 \left[\frac{\sin(2\pi/n)}{(2\pi/n)} \right].$$

So the area of a hexagon inscribed in a circle is $(3\sqrt{3}/2)r^2$, twice that of a triangle inscribed in the same circle. Tessellations with near-triangular cells will thus combine information from orientations which are $\sqrt{2}$ times as far from the average as do tessellation using near-hexagonal cells.

If cells occur in a regular pattern, the relationship of a cell to its neighbors will be the same for all cells. Such arrangements are to be preferred. Unfortunately, it is not possible to simultaneously satisfy the criteria listed above.

A simple tessellation consists of a division into latitude bands, each of which is then further divided along longitudinal strips (Figure 15). The cells could be made more nearly equal in area by having fewer at higher latitudes, or by making the latitude bands wider there, or both. One advantage of this scheme is that it makes it easy to compute to which cell a particular surface normal should be assigned. Still, this arrangement does not come

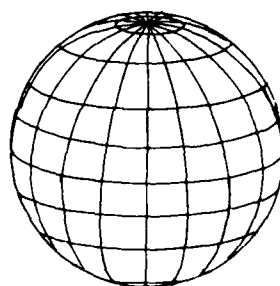


Figure 15. The Gaussian sphere can be divided into cells along meridians and lines of longitude. The resulting cells do not have the same areas however and only align with each other for certain rotations about the axis through the poles.

close to satisfying the criteria stated above. In particular, the cells are brought into alignment only for a few rotations about the axis of the globe. Rotations about any other axis can not bring the cells into alignment.

5.1. Tessellations Based on Regular Polyhedra

Better tessellations may be found by projecting regular polyhedra onto the unit sphere after bringing their center to the center of the sphere [27]. Regular polyhedra are uniform and have faces which are all of one kind of regular polygon (They are also called the Platonic solids). [19, 20, 28-32] The vertices of a regular polyhedron are congruent. A division obtained by projecting a regular polyhedron has the desirable property that the resulting cells all have the same shape and area. Also, all cells have the same geometric relationship to their neighbors. In the case of the dodecahedron, the cells are even fairly well rounded. The dodecahedron, however, has only twelve cells (Figure 16a). Even the icosahedron, with twenty triangular cells, provides too coarse a sampling of orientations (Figure 16b). Furthermore, its cells are not well rounded. Unfortunately there are only five regular solids (tetrahedron, hexahedron, octahedron, dodecahedron, and icosahedron).

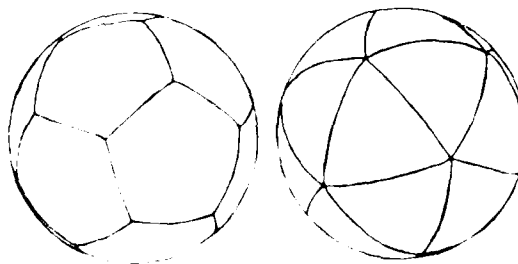


Figure 16. Tessellations of the Gaussian sphere using (a) the regular dodecahedron and (b) the regular icosahedron.

One can go a little further by considering semi-regular polyhedra. A semi-regular polyhedron has regular polygons

as faces, but the faces are not all of the same kind [19, 20, 28-32] (They are also called the Archimedean polyhedra). As for regular polyhedra, the vertices are congruent. There can be either two or three different types of faces and these have different areas. An illustration of a tessellation using a semi-regular polyhedron is provided by a soccer ball (Figure 17a). It is based on the truncated icosahedron, a semi-regular polyhedron which has 12 pentagonal faces and 20 hexagonal faces.

Unfortunately there are only 13 semi-regular polyhedra (The five truncated regular polyhedra, cuboctahedron, icosidodecahedron, snub cuboctahedron, snub icosidodecahedron, truncated cuboctahedron, rhombicuboctahedron, truncated icosidodecahedron, and the rhombicosidodecahedron). Overall, these objects do not provide us with fine enough tessellations. The snub icosidodecahedron has the largest number of faces, but each of its 80 triangles is much smaller than each of its 12 pentagons.

The edges of a semi-regular polyhedron are all the same length. One consequence of this is that the different types of faces have different areas. The area of a regular polygon of n sides and edge-length e equals

$$\frac{ne^2}{4 \tan(\pi/n)},$$

so it is very roughly proportional to n^2 . This is a problem generally with semi-regular polyhedra. It is sometimes possible to derive a new polyhedron which has the same adjacency relationships between faces as a given semi-regular polyhedron but also has faces of equal area. The shapes of some of these faces then are no longer regular, however.

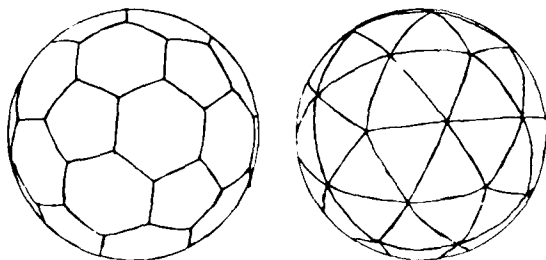


Figure 17. Tessellation of the Gaussian sphere using (a) the truncated icosahedron and (b) the pentakis dodecahedron.

If we desire a finer subdivision still, we can consider splitting each face of a given tessellation further into triangular facets. If, for example, we split each pentagonal face of a dodecahedron into five equal triangles we obtain a pentakis dodecahedron with 60 faces (Figure 17b). This happens to be the dual of the truncated icosahedron, discussed above. If we apply this method instead to the truncated icosahedron we construct an object with 180 faces. This object, as well as the pentakis dodecahedron,

form suitable bases for further subdivision, as we shall show later.

To see how fine a division we might need, let us calculate the angular spread of surface normals which map into a particular cell. If there are n equal cells, then each one will have area

$$A = (4\pi)/n,$$

since the total area of the unit sphere is 4π (This area equals the solid angle of the cone formed by the cell when connected to the center of the sphere). The shape which minimizes the angular spread for given surface area is the circular disc. The area of a circular disc on the unit sphere is

$$A = 2\pi(1 - \cos \theta),$$

where θ is the half-angle of the cone formed by the disc when connected to the center of the sphere. If θ is small, the area can be approximated by

$$A \approx \pi\theta^2.$$

Thus if there are many cells and if they could be made circular, the angular spread would be

$$\theta \approx 2/\sqrt{n}.$$

The best we can hope for, however, are near-hexagonal cells. The area of a hexagon inscribed in a circle of radius r is $(3\sqrt{3}/2)r^2$, as already mentioned. The area of the circle, πr^2 , is about 20% more. So a hexagonal shape has a spread which is

$$\sqrt{\frac{2\pi}{3\sqrt{3}}} = 1.0996\dots$$

as large as that of a circular shape of equal area. A lower bound on the angular spread for a tessellation with n cells then is

$$\theta = \sqrt{\frac{4\pi}{3\sqrt{3}n}} = \frac{2.1993\dots}{\sqrt{n}}.$$

For $n = 60$, for example, the spread is at greater than 16.2° . One should also remember that the spread for triangular cells is even more, namely $\sqrt{2}$ times that for hexagonal cells.

5.2. Geodesic Domes

To proceed further, we can divide the triangular cells into four smaller triangles according to the well known geodesic dome constructions [27, 30, 33]. We attain high resolution by relenting on several of the criteria given above (Figure 18). Specifically, the cells of a geodesic tessellation do not all have the same area and shape. The cells are also not compact, being shaped like (irregular) triangles. The duals of geodesic domes are better in this respect, since they have facets that are mostly (irregular) hexagons, with a dozen (regular) pentagons thrown in. Tessellations of arbitrary fineness can be constructed in this fashion. The pentakis dodecahedron is a good starting point for a geodesic division, as is the object constructed earlier

from the truncated icosahedron by dividing the faces into triangles.

Each of the edges of the triangular cells of the original polyhedron are divided into f sections, where f is called the frequency of the geodesic division. The result is that each face is divided into f^2 (irregular) triangles. Tessellations where the frequency is a power of two are particularly well suited to the method suggested here, as we see next.

One has to be able to efficiently compute to which cell a particular surface normal belongs. In the case of the tessellations derived from regular polyhedra, one first computes the dot-product of the given unit vector and the vector to the center of each cell (These reference vectors correspond to the vertices of the dual of the original regular polyhedron). This gives one the cosine of the angle between the two. The closest reference vector is the one which gives the largest dot-product. The given vector is then assigned to the cell corresponding to that reference vector.

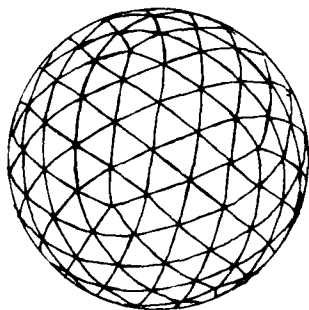


Figure 18. Tessellation of the Gaussian sphere using a frequency four geodesic tessellation based on the icosahedron (There are $16 \times 20 = 320$ faces).

In the case of a geodesic dome, it is possible to proceed hierarchically, particularly if the frequency is a power of two. The geodesic dome is based on some regular polyhedron. The appropriate facet of this polyhedron is found as above. Next, one determines into which of the triangles of the first division of this facet the given unit normal falls. This can be done by considering which dot-product has the *second* largest value. No new dot-products need to be computed. The process is then repeated with the four triangles into which this facet is divided, and so on. In practice, lookup table methods can be used, which, while not exact, are very quick.

Let the area occupied by one of the cells on the Gaussian sphere be ω (in the case of the icosahedron $\omega = 4\pi/20$). The expected number of surface normals mapped into a cell equals

$$\rho \omega |\bar{G}|,$$

for a convex object, where \bar{G} is the average of $G(\xi, \eta)$ over the cell.

It is clear that the extended Gaussian image can be computed locally. One simply counts the number of surface normals that belong in each cell. The expression for the Gaussian curvature, on the other hand, includes first and second partial derivatives of the surface function. In practice, estimates of derivatives are unreliable, because of noise. It is important therefore that the extended Gaussian image can be computed *without* estimating the derivatives.

The values in the cells can be thought of as an orientation histogram. It has recently been brought to my attention that this is analogous to a scheme used for histogramming directions of dendrites on neurons [34].

The result can be displayed graphically using normal vectors on each of the cells to represent the weight of the accumulated surface normals. A frequency two sub-division of the pentakis dodecahedron provides 240 cells, enough for most practical purposes (Figure 19). The angular spread in this case is about 11.5° . An alternative way to present the extended Gaussian image graphically is by means of a grey-level image where brightness in each cell is proportional to the count. The surface of the Gaussian sphere may be projected stereographically instead of orthographically in order to preserve the shapes of the cells. Their areas will be scaled unequally however.

A further refinement of the orientation histogram has us store the sum of the vectors, scaled according to the area of the corresponding patch, rather than just the sum of the areas of the patches. This requires three times as much memory space, but provides more accuracy. In fact, in the case of polyhedra, this representation is exact.

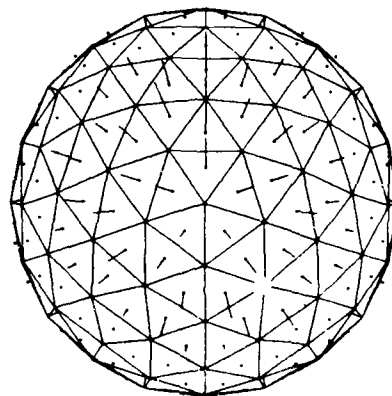


Figure 19. Orientation Histogram collected on a geodesic dome derived from the pentakis dodecahedron (There are $12 \times 5 \times 4 = 240$ faces). This is a discrete approximation of the extended Gaussian image. The length of the vector attached to the center of a cell is proportional to the number of surface normals on the surface of the original object which have orientations falling within the range of directions spanned by that cell.

6. Solids of Revolution

In the case of the surface of a solid of revolution, the Gaussian curvature is rather easy to determine. The solid of revolution can be produced by rotating a (planar) generating curve about an axis (Figure 20). Let the generating curve be specified by the perpendicular distance from the axis, $r(s)$, given as a function of arc length s along the curve. Let θ be the angle of rotation around the axis. Now consider the Gaussian sphere positioned so that its axis is aligned with the axis of the solid of revolution. Let ξ be the longitude and η be the latitude on the Gaussian sphere.

We can let ξ correspond to θ . That is, a point on the object produced when the generating curve has rotated through an angle θ has a surface normal that lies on the Gaussian sphere at a point with longitude $\xi = \theta$.

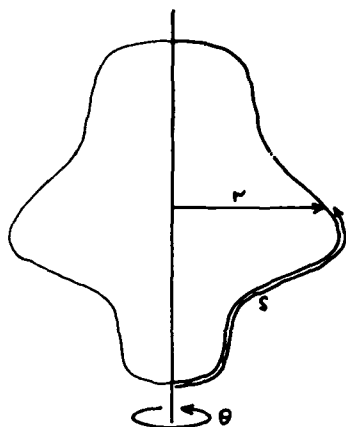


Figure 20. A solid of revolution can be generated by rotating a curve around an axis. The curve can be specified by giving the distance from the axis as a function of the arc length along the curve.

6.1. Gaussian Curvature of Solid of Revolution

Consider a small patch on the Gaussian sphere lying between ξ and $\xi + \delta\xi$ in longitude and between η and $\eta + \delta\eta$ in latitude. Its area is

$$\cos \eta \delta\xi \delta\eta.$$

We need only determine the area of the corresponding patch on the object. It is

$$r \delta\theta \delta s,$$

where δs is the change in arc distance along the generating curve corresponding to the change $\delta\eta$ in surface orientation. The Gaussian curvature is the limit of the ratio of the two areas as they tend to zero. That is,

$$K = \lim_{\substack{\delta\xi \rightarrow 0 \\ \delta\eta \rightarrow 0}} \frac{\cos \eta \delta\xi \delta\eta}{r \delta\theta \delta s} = \lim_{\delta\eta \rightarrow 0} \frac{\cos \eta \delta\eta}{r \delta s} = \frac{\cos \eta}{r} \frac{d\eta}{ds},$$

since $\delta\xi = \delta\theta$. The curvature of the generating curve, κ_G , is just the rate of change of direction with arc length along it [22-24]. So

$$\frac{d\eta}{ds} = \kappa_G$$

and hence

$$K = \frac{\kappa_G \cos \eta}{r}.$$

It is easy to see that (Figure 21), $\sin \eta = -r_s$, where r_s is the partial derivative of r with respect to s . Differentiating with respect to s we get

$$\cos \eta \frac{d\eta}{ds} = \frac{d}{ds}(-r_s) = -r_{ss},$$

and so we obtain the simple formula

$$K = -\frac{r_{ss}}{r}.$$

In the case of a sphere of radius R , for example, we have $r = R \cos(s/R)$ for $-(\pi/2)R < s < +(\pi/2)R$. Thus $r_{ss} = -(r/R^2)$ and $K = 1/R^2$.

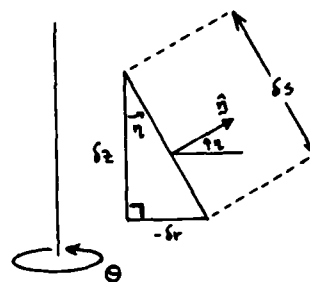


Figure 21. This figure shows the relationships between infinitesimal increments in arc length along the curve, distance from the axis of rotation and distance along the axis of rotation.

For some purposes it is more useful to express the radius r as a function of the distance along the axis, rather than as a function of the distance along the curve. Let the distance along the axis be denoted by z . It is easy to see that (Figure 21), $\tan \eta = -r_z$, and so, differentiating with respect to s ,

$$\sec^2 \eta \frac{d\eta}{ds} = \frac{d}{ds}(-r_z) = -r_{zs} \frac{dz}{ds},$$

where from the figure we see that $\cos \eta = z_s$, so that

$$\kappa_G \cos \eta = \frac{d\eta}{ds} \cos \eta = -r_{zs} \cos^3 \eta,$$

and finally

$$K = -\frac{r_{zz}}{r(1+r_z^2)^2},$$

since

$$\sec^2 \eta = 1 + r_z^2.$$

6.2. Alternate Derivation of Gaussian Curvature of a Solid of Revolution (*)

We first need to review Meusnier's theorem [22-24]. Consider a normal section of a surface at a particular point. It is obtained by cutting the surface with one of the planes including the local normal (Figure 22). Suppose that the curvature of the curve in which the surface cuts this plane is κ_N . Now imagine tilting the plane away from the normal by an angle η (using the local tangent as an axis to rotate about). The new plane will cut the surface in a curve with higher curvature. In fact, it can be shown that the new curve has curvature

$$\kappa_N / \cos \eta.$$

It is easy to see this in the case of a sphere, since a plane including the center cuts the sphere in a great circle, while an inclined plane cuts it in a small circle of radius proportional to the cosine of the angle of inclination.

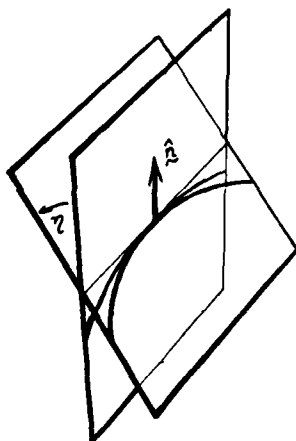


Figure 22. The curvature of the curve obtained by cutting a surface using an inclined plane is greater than that obtained by cutting it using a plane which includes the surface normal. Meusnier's theorem tells us that the ratio of the two curvatures is equal to the cosine of the angle between the two planes.

Now, let us return to the surface of revolution. It is not hard to show that one of the principal curvatures at a point on the surface will correspond to a cut through the surface by a plane which includes the axis of revolution. The curve obtained in this way is just the generating curve of the solid of revolution. So one of the two principal curvatures is equal to the curvature κ_G of the generating curve at the corresponding point.

Now consider a plane perpendicular to the axis of revolution through the same surface point (Figure 23). It cuts the surface in a circle. The curvature in this plane equals $(1/r)$, where r is the radius of the solid of revolution at that point. This horizontal plane, however, is *not* a normal section. Suppose that the normal makes an angle η

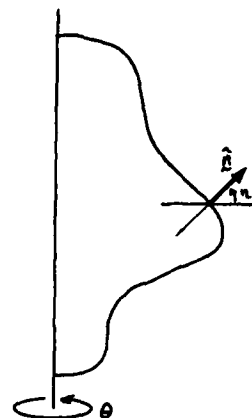


Figure 23. If the solid of revolution is cut by a plane perpendicular to the axis of rotation, a circle is obtained. The curvature in this plane is just the inverse of the distance of the surface from the axis. The curvature of the corresponding normal section can be obtained using Meusnier's theorem.

relative to this plane (The local tangent plane also makes an angle η relative to the axis of revolution). Now construct the plane including the local normal which intersects the horizontal plane in a line perpendicular to the axis. This plane will be inclined η relative to the one we have just studied. It also produces the second principal normal section sought after. By Meusnier's theorem we see that the curvature of the curve found in this normal section is $\kappa_N = (1/r) \cos \eta$. Finally, the Gaussian curvature is found by multiplication to be

$$K = \frac{\kappa_G \cos \eta}{r}.$$

In the case of a sphere of radius R , for example, we have $r = R \cos \eta$ and $\kappa_G = 1/R$, so that $K = 1/R^2$, as expected.

To make this result more usable, erect a coordinate system with the z -axis aligned with the axis of revolution. The generating curve is given as $r(z)$. Let the first and second derivatives of r with respect to z be denoted by r_z and r_{zz} respectively. It is easy to see that (Figure 21), $\tan \eta = r_z$, so that

$$\cos \eta = \frac{1}{\sqrt{1 + r_z^2}}.$$

Furthermore,

$$\kappa_G = -\frac{r_{zz}}{(1 + r_z^2)^{3/2}},$$

so that finally,

$$K = -\frac{r_{zz}}{r(1 + r_z^2)^2}.$$

In order to use this result in deriving extended Gaussian images it is necessary to identify points on the surface with

points on the Gaussian sphere. Suppose that we introduce a polar angle θ such that

$$x = r \cos \theta \quad \text{and} \quad y = r \sin \theta.$$

Then a unit normal to the surface is given by

$$\frac{(\cos \theta, \sin \theta, -r_z)^T}{\sqrt{1 + r_z^2}}.$$

Equating this to the unit normal on the Gaussian sphere,

$$(\cos \xi \cos \eta, \sin \xi \cos \eta, \sin \eta)^T,$$

we get

$$\xi = \theta \quad \text{and} \quad \tan \eta = -r_z.$$

6.3. Extended Gaussian Image of a Torus

As an illustration we will now determine the extended Gaussian image of a torus. Let the torus have major axis R and minor axis ρ (Figure 24). A point on the surface can be identified by θ and s , where θ is the angle around the axis of the torus, while s is the arc length along the surface measured from the plane of symmetry. Then,

$$r = R + \rho \cos(s/\rho),$$

and

$$r_{ss} = -(1/\rho) \cos(s/\rho),$$

so that

$$K = -\frac{r_{ss}}{r} = \frac{1}{\rho R + \rho \cos(s/\rho)} \cos(s/\rho).$$

Two points, P and P' (Figure 24), separated by π in θ , have the same surface orientation on the torus. The surface normal at one of these places points away from the axis of rotation, while it points towards the axis at the other place. Accordingly, two points on the object,

$$(\theta, s) = (\xi, \rho \eta) \quad \text{and} \quad (\theta, s) = (\xi + \pi, \rho(\pi - \eta)),$$

correspond to the point (ξ, η) on the Gaussian sphere. The curvatures at these two points have opposite signs

$$K_+ = +\frac{1}{\rho R + \rho \cos \eta} \quad \text{and} \quad K_- = -\frac{1}{\rho R - \rho \cos \eta}.$$

The torus is not a convex object, so more than one point on its surface contributes to a given point on the extended Gaussian image. If we add the absolute values of the inverses of the curvature we get

$$G(\eta, \xi) = \frac{1}{K_+} + \frac{1}{K_-} = 2R\rho \sec \eta.$$

If we had added the inverses algebraically instead we would have obtained

$$\frac{1}{K_+} + \frac{1}{K_-} = 2\rho^2,$$

which is twice the result for a sphere of radius ρ .

The same results could have been found using

$$K = \frac{\kappa_G \cos \eta}{r},$$

since $\kappa_G = -1/\rho$ and $r = R \pm \rho \cos \eta$, so that

$$K = \frac{\pm \cos \eta}{\rho(R \pm \rho \cos \eta)}.$$

The extended Gaussian image of a torus has singularities at the poles. These correspond to the two rings on which the torus would rest if it were dropped onto a plane. All of the points on one of these rings have the same surface orientation.

We can think of the Gaussian sphere as covered by two sheets of a Riemann surface, one corresponding to the inner half of the torus, closer to its axis of symmetry, the other corresponding to the outer half. The two sheets are connected to one another at the poles, branch points corresponding to the two rings mentioned above. There the Gaussian curvature changes sign.

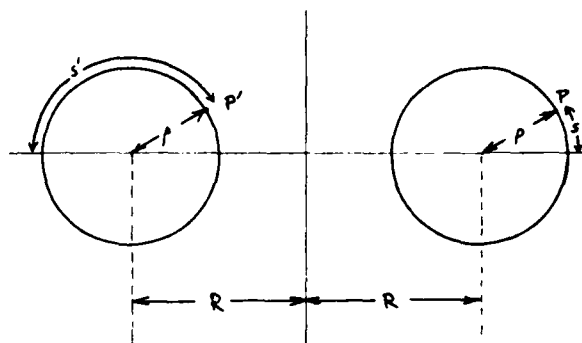


Figure 24. A torus obtained by spinning a circle around an axis. The resulting object is not convex. Its extended Gaussian image can be computed nevertheless.

We may also note at this point that all tori with the same surface area, $(4\pi^2\rho R)$, have the same extended Gaussian image.

6.4. The Unique Convex Object with $G(\xi, \eta) = 2 \sec \eta$ (*)

While all tori with surface area $4\pi^2$ have the same extended Gaussian image,

$$G(\xi, \eta) = 2 \sec \eta,$$

there is only one convex object which has that extended Gaussian image. It is a solid of revolution since $G(\xi, \eta)$ is independent of ξ . So we have on the one hand

$$K = 1/2 \cos \eta,$$

and on the other hand

$$K = \frac{\kappa_G \cos \eta}{r},$$

so that

$$\kappa_G = r/2.$$

The equation states that the curvature of the generating curves varies linearly with the distance from the axis of

rotation. This deceptively simple equation represents a non-linear second order differential equation for r in terms of z since

$$\kappa_G = -\frac{r_{zz}}{(1+r_z^2)^{3/2}},$$

so that

$$r_{zz} = -\frac{r}{2}(1+r_z^2)^{3/2}.$$

Now

$$\frac{d}{dz} \frac{1}{\sqrt{1+r_z^2}} = -\frac{r_z r_{zz}}{(1+r_z^2)^{3/2}}$$

and

$$\frac{d}{dz} \frac{r^2}{4} = \frac{r r_z}{2},$$

so that

$$\frac{d}{dz} \frac{1}{\sqrt{1+r_z^2}} = \frac{d}{dz} \frac{r^2}{4}$$

or

$$\frac{1}{\sqrt{1+r_z^2}} = \frac{r^2 + c^2}{4},$$

where c^2 is a constant of integration. We now have reduced the problem to a non-linear first order differential equation for r in terms of z . If the object is to be convex and smooth at its poles, we expect $r_z \rightarrow \infty$ as $r \rightarrow 0$. Thus $c = 0$. Next note that the term on the left equals $\cos \eta$. So we also have

$$\cos \eta = \frac{r^2}{4}$$

or, using an earlier expression for κ_G ,

$$\kappa_G = -\sqrt{\cos \eta}.$$

This is an implicit equation for the curve of least energy [35]! The curve of least energy is the curve which minimizes the integral of the square of the curvature κ_G . It can be solved for z in terms of r to yield

$$z = \sqrt{2}[2E(\cos^{-1}(r/2), 1/\sqrt{2}) - F(\cos^{-1}(r/2), 1/\sqrt{2})],$$

where E and F are incomplete elliptic integrals. If we let s be the arc length along the curve we can also write the solution in Whewell form

$$s = \sqrt{2}F(\cos^{-1} \sqrt{\cos \eta}, 1/\sqrt{2}),$$

or Césaro form

$$s = \sqrt{2}F(\cos^{-1}(-\kappa_G), 1/\sqrt{2}).$$

The length of the curve from the pole to the equator is

$$\sqrt{2}K(1/\sqrt{2}) = \sqrt{2}K(\sin(\pi/4)) = \frac{\Gamma(1/4)^2}{2\sqrt{2\pi}},$$

where K is the complete elliptic integral of the first kind, and Γ is the gamma-function [24]. The height from equator to the pole is

$$W = \frac{(2\pi)^{3/2}}{\Gamma(1/4)^2}$$

while the maximum radius is

$$H = 2.$$

The minimum radius of curvature equals one, so that a circle tangent at the outermost point is also tangent at the origin [35]. This circle, when rotated about the vertical axis, produces a torus with the same extended Gaussian image (Figure 25). Both objects have total surface area $4\pi^2$.

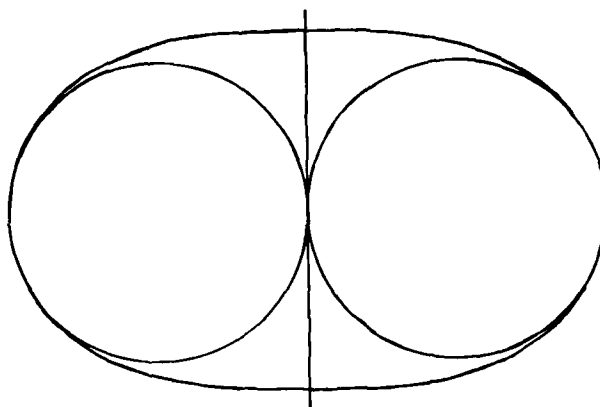


Figure 25. The unique convex object with the same extended Gaussian image as a torus has an interesting shape. It is a solid of revolution whose generating curve is the curve of least energy. This is the shape which a uniform bar constrained to pass through two points in space with given orientation will adopt.

7. Gaussian Curvature in the General Case

When the object is not a solid of revolution we need to work a little harder to obtain the Gaussian curvature. Let $x = x(u, v)$, $y = y(u, v)$, and $z = z(u, v)$ be parametric equations for points on a given surface. Let $\mathbf{r} = (x, y, z)^T$ be a vector to a point on the surface. Then

$$\mathbf{r}_u = \frac{\partial \mathbf{r}}{\partial u} \quad \text{and} \quad \mathbf{r}_v = \frac{\partial \mathbf{r}}{\partial v}$$

are two tangents to the surface, as already noted earlier. The cross product of these two vectors,

$$\mathbf{n} = \mathbf{r}_u \times \mathbf{r}_v,$$

will be perpendicular to the local tangent plane (Figure 14). The length of this normal vector squared equals

$$n^2 = \mathbf{n} \cdot \mathbf{n} = (\mathbf{r}_u \cdot \mathbf{r}_u)(\mathbf{r}_v \cdot \mathbf{r}_v) - (\mathbf{r}_u \cdot \mathbf{r}_v)^2$$

since $(\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{d}) = (\mathbf{a} \cdot \mathbf{c})(\mathbf{b} \cdot \mathbf{d}) - (\mathbf{a} \cdot \mathbf{d})(\mathbf{b} \cdot \mathbf{c})$. A unit vector $\hat{\mathbf{n}} = \mathbf{n}/n$ can be computed using this result.

7.1. Gaussian Curvature from Variation in Normals (*)

The Gaussian curvature is the limit of the ratio of the area of a patch on the Gaussian sphere to the area of the corresponding patch on the surface, as the area shrinks to zero. Consider an infinitesimal triangle formed by the three points on the surface corresponding to (u, v) , $(u + \delta u, v)$, and $(u, v + \delta v)$. The lengths of two sides of this triangle are

$$|r_u| \delta u \quad \text{and} \quad |r_v| \delta v$$

while the sine of the angle between these sides equals

$$\frac{|r_u \times r_v|}{|r_u| |r_v|}$$

so that an outward normal with size equal to the area of the triangle is given by

$$\frac{1}{2} (r_u \times r_v) \delta u \delta v = \frac{1}{2} n \delta u \delta v.$$

To determine the area of the corresponding triangular patch on the Gaussian sphere we need to find the unit surface normals at the three points. The unit surface normals will be

$$\hat{n}, \quad \hat{n} + \hat{n}_u \delta u, \quad \text{and} \quad \hat{n} + \hat{n}_v \delta v$$

if we ignore terms of higher order in δu and δv . Here \hat{n}_u and \hat{n}_v are the partial derivatives of \hat{n} with respect to u and v . Note that \hat{n}_u and \hat{n}_v are perpendicular to \hat{n} . The area of the patch on the Gaussian sphere equals the magnitude of

$$\frac{1}{2} (\hat{n}_u \times \hat{n}_v) \delta u \delta v$$

by reasoning similar to that used in determining the area of the original patch on the given surface. We need to find \hat{n}_u and \hat{n}_v to compute this area. Now

$$\hat{n}_u = \frac{\partial \hat{n}}{\partial u} = \frac{n n_u - n n_u}{n^2}.$$

From $n^2 = n \cdot n$ we get

$$n n_u = n \cdot n_u,$$

so that

$$\hat{n}_u = \frac{(n \cdot n) n_u - (n \cdot n_u) n}{n^3} = \frac{(n \times n_u) \times n}{n^3},$$

and

$$\hat{n}_v = \frac{(n \cdot n) n_v - (n \cdot n_v) n}{n^3} = \frac{(n \times n_v) \times n}{n^3},$$

since $(a \times b) \times c = (a \cdot c)b - (b \cdot c)a$. Then

$$\hat{n}_u \times \hat{n}_v = \frac{n^2}{n^6} [(n \cdot n)(n_u \times n_v) + (n \cdot n_u)(n_v \times n) + (n \cdot n_v)(n \times n_u)]$$

or

$$\hat{n}_u \times \hat{n}_v = \frac{1}{n^4} [n n_u n_v] n$$

since $[a b c] p = (a \cdot p)(b \times c) + (b \cdot p)(c \times a) + (c \cdot p)(a \times b)$, where $[a b c] = (a \times b) \cdot c$.

This shows that the patch on the Gaussian sphere has

the same orientation as the patch on the surface, as it should. An outward pointing normal of size equal to the area is given by

$$\frac{1}{2} \frac{1}{n^4} [n n_u n_v] n \delta u \delta v.$$

The ratio of the two areas, the Gaussian curvature, then just

$$K = \frac{[n n_u n_v]}{n^4}.$$

Now

$$n = r_u \times r_v,$$

so that

$$n_u = r_{uu} \times r_v + r_u \times r_{uv},$$

and

$$n_v = r_{uv} \times r_v + r_u \times r_{vv}.$$

using $(a \times b) \times (c \times d) = [a b d]c - [a b c]d$ or $(a \times b) \times (c \times d) = [a c d]b - [b c d]a$ we get

$$\begin{aligned} n_u \times n_v &= -[r_{uu} r_v r_{vv}] r_v + [r_{uu} r_v r_{vv}] r_u \\ &\quad - [r_{uu} r_v r_u] r_{vv} \\ &\quad + [r_u r_{uv} r_v] r_{uv} \\ &\quad + [r_u r_{uv} r_{vv}] r_u \end{aligned}$$

so that

$$[n n_u n_v] = n \cdot (n_u \times n_v) = [r_u r_v r_{uu}][r_u r_v r_{vv}] - [r_u r_v r_{uv}]^2$$

, and finally

$$K = \frac{[r_u r_v r_{uu}][r_u r_v r_{vv}] - [r_u r_v r_{uv}]^2}{|r_u \times r_v|^4}.$$

This result can be used to derive the expression for curvature of a solid of revolution in a more rigorous fashion.

7.2. Fundamental Forms of a Surface (*)

Let, as before,

$$\hat{n} = \frac{r_u \times r_v}{|r_u \times r_v|},$$

be the unit surface normal vector. The first fundamental form of a surface gives the square of the element of distance as [24]

$$ds^2 = |dr|^2 = E(u, v) du^2 + 2F(u, v) du dv + G(u, v) dv^2.$$

The second fundamental form of a surface gives the normal curvature using the equation [24]

$$-dr \cdot d\hat{n} = L(u, v) du^2 + 2M(u, v) du dv + N(u, v) dv^2.$$

The coefficients can be expressed in terms of derivatives of r as follows,

$$E = r_u \cdot r_u, \quad F = r_u \cdot r_v, \quad \text{and} \quad G = r_v \cdot r_v$$

and

$$L = r_u \cdot \hat{n}_u, \quad M = r_u \cdot \hat{n}_v = r_v \cdot \hat{n}_u, \quad \text{and} \quad N = r_v \cdot \hat{n}_v,$$

or,

$$L = \frac{[r_u r_v r_{uu}]}{\sqrt{EG - F^2}}, \quad M = \frac{[r_u r_v r_{uv}]}{\sqrt{EG - F^2}}, \quad \text{and}$$

$$N = \frac{[r_u r_v r_{vv}]}{\sqrt{EG - F^2}}$$

so that [24]

$$K = \frac{LN - M^2}{EG - F^2}.$$

Finally, if the surface is given as $z(x, y)$, the above reduces to the familiar

$$K = \frac{z_{xx}z_{yy} - z_{xy}^2}{(1 + z_x^2 + z_y^2)^2}.$$

7.3. Application of the General Formula to the Ellipsoid (*)

In the case of the ellipsoid we have, as discussed before,

$$\begin{aligned} \mathbf{r} &= (a \cos \theta \cos \phi, b \sin \theta \cos \phi, c \sin \phi)^T, \\ \mathbf{r}_\theta &= (-a \sin \theta \cos \phi, b \cos \theta \cos \phi, 0)^T, \\ \mathbf{r}_\phi &= (-a \cos \theta \sin \phi, -b \sin \theta \sin \phi, c \cos \phi)^T, \end{aligned}$$

and

$$\begin{aligned} \mathbf{r}_{\theta\theta} &= (-a \cos \theta \cos \phi, -b \sin \theta \cos \phi, 0)^T, \\ \mathbf{r}_{\theta\phi} &= (a \sin \theta \sin \phi, -b \cos \theta \sin \phi, 0)^T, \\ \mathbf{r}_{\phi\phi} &= (-a \cos \theta \cos \phi, -b \sin \theta \cos \phi, -c \sin \phi)^T. \end{aligned}$$

A surface normal can be found by taking cross products, $\mathbf{n} = \mathbf{r}_\theta \times \mathbf{r}_\phi = (bc \cos \theta \cos \phi, ca \sin \theta \cos \phi, ab \sin \phi)^T \cos \phi$, and the coefficients of the first fundamental form are,

$$\begin{aligned} E &= \mathbf{r}_\theta \cdot \mathbf{r}_\theta = (a^2 \sin^2 \theta + b^2 \cos^2 \theta) \cos^2 \phi, \\ F &= \mathbf{r}_\theta \cdot \mathbf{r}_\phi = (a^2 + b^2) \sin \theta \cos \theta \sin \phi \cos \phi, \\ G &= \mathbf{r}_\phi \cdot \mathbf{r}_\phi = (a^2 \cos^2 \theta + b^2 \sin^2 \theta) \sin^2 \phi + c^2 \cos^2 \phi. \end{aligned}$$

Hence

$$EG - F^2 = [(bc \cos \theta \cos \phi)^2 + (ca \sin \theta \cos \phi)^2 + (ab \sin \phi)^2] \cos^2 \phi.$$

To compute the coefficients of the second fundamental form we need,

$$\begin{aligned} [\mathbf{r}_\theta \mathbf{r}_\phi \mathbf{r}_{\theta\theta}] &= \mathbf{n} \cdot \mathbf{r}_{\theta\theta} = -abc \cos^3 \phi, \\ [\mathbf{r}_\theta \mathbf{r}_\phi \mathbf{r}_{\theta\phi}] &= \mathbf{n} \cdot \mathbf{r}_{\theta\phi} = 0, \\ [\mathbf{r}_\theta \mathbf{r}_\phi \mathbf{r}_{\phi\phi}] &= \mathbf{n} \cdot \mathbf{r}_{\phi\phi} = -abc \cos \phi, \end{aligned}$$

and so

$$[\mathbf{r}_\theta \mathbf{r}_\phi \mathbf{r}_{\theta\theta}][\mathbf{r}_\theta \mathbf{r}_\phi \mathbf{r}_{\phi\phi}] - [\mathbf{r}_\theta \mathbf{r}_\phi \mathbf{r}_{\theta\phi}]^2 = (abc \cos^2 \phi)^2.$$

Finally then,

$$LN - M^2 = \frac{(abc \cos^2 \phi)^2}{EG - F^2},$$

and

$$K = \frac{LN - M^2}{EG - F^2} = \left[\frac{abc}{(bc \cos \theta \cos \phi)^2 + (ca \sin \theta \cos \phi)^2 + (ab \sin \phi)^2} \right]^2.$$

This result was used earlier in the discussion of the extended Gaussian image of the ellipsoid.

8. Summary and Conclusions

We have defined the extended Gaussian image, discussed its properties and given examples. Methods for determining the extended Gaussian images of polyhedra, solids of revolution and smoothly curved objects in general were shown. The orientation histogram, a discrete approximation of the extended Gaussian image, was described along with a variety of ways of tessellating the sphere. Machine vision methods for obtaining the surface orientation information required to build an orientation histogram are discussed elsewhere [1, 3-8]. Extended Gaussian images based on object models can be matched with those derived from experimental data. The application of extended Gaussian images to object recognition and, more importantly, to finding the attitude in space of an object, are discussed in a recent article [17].

9. Acknowledgment

I wish to thank Eric Grimson and Tomás Lozano-Pérez who made a number of very helpful suggestions after reading a draft of this paper.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Office of Naval Research under Office of Naval Research contract N00014-77-C-0389 and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505.

10. References

- [1] W. E. L. Grimson, *From Images to Surfaces*, M. I. T. Press, 1981.
- [2] B. K. P. Horn, "Sequins and Quills—representations for surface topography," M.I.T. A.I. Laboratory Memo 536, May 1979.
- [3] R. J. Woodham, "Photometric Stereo: A Reflectance Map Technique for Determining Surface Orientation from a Single View," *Image Understanding Systems and Industrial Applications*, Proc. S. P. I. E. 22nd Annual Technical Symposium, vol. 155, pp. 136-143, August 1978.

- [4] B. K. P. Horn, R. J. Woodham, and W. M. Silver, "Determining Shape and Reflectance using Multiple Images," M.I.T. A.I. Laboratory Memo 490, August 1978.
- [5] R. J. Woodham, "Photometric Method for Determining Surface Orientation from Multiple Images," *Optical Engineering*, vol. 19, no. 1, Jan/Feb. 1980, pp. 139-144.
- [6] W. M. Silver, *Determining Shape and Reflectance Using Multiple Images*, S. M. Thesis, Department of Electrical Engineering and Computer Science, M.I.T., June 1980.
- [7] Katsushi Ikeuchi, "Determining Surface Orientation of Specular Surfaces Using the Photometric Stereo Method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, no. 6, November 1981, pp. 661-669.
- [8] E. N. Coleman and R. Jain, "Obtaining 3-Dimensional Shape of Textured and Specular Surfaces using Four-Source Photometry," *Computer Graphics and Image Processing*, vol. 18, no. 4, pp. 309-328, 1982.
- [9] D. A. Smith, "Using Enhanced Spherical Images," M.I.T. A.I. Laboratory Memo No. 530, May 1979.
- [10] R. Bajcsy, "Three-Dimensional Scene Analysis," *Proc. 5th International Pattern Recognition Conf.*, Miami, December 1980, pp. 1064-1074.
- [11] Katsushi Ikeuchi, "Recognition of Objects Using the Extended Gaussian Image," *Proc. IJCAI-81*, Vancouver, B. C., August 1981, pp. 595-600.
- [12] C. Dane and R. Bajcsy, "Three-Dimensional Segmentation using the Gaussian Image and Spatial Information," *Proc. IEEE Computer Society Conf. on Pattern Recognition and Image Processing*, Dallas, August 1981, pp. 54-56.
- [13] D. H. Ballard and D. Sabbah, "Detecting Object Orientation from Surface Normals," *Proc. International Pattern Recognition Conf.*, München, December 1981, pp. 63-67.
- [14] P. Brou, "Finding Objects in Depth Maps," Ph. D. Thesis, M. I. T. Department of Electrical Engineering and Computer Science, September 1983.
- [15] Katsushi Ikeuchi, "Determining the Attitude of an Object from a Needle Map using the Extended Gaussian Image," M.I.T. A.I. Laboratory Memo No. 714, April 1983.
- [16] Katsushi Ikeuchi, B. K. P. Horn, Shigemi Nagata, T. Callahan, and O. Feingold, "Picking an Object from a Pile of Objects," M.I.T. A.I. Laboratory Memo No. 726, May 1983.
- [17] B. K. P. Horn and Katsushi Ikeuchi, "Mechanically Manipulating Randomly Oriented Parts," *Scientific American*, August 1984.
- [18] H. Minkowski, "Allgemeine Lehrsätze über die konvexen Polyeder," *Nachrichten von der Königlichen Gesellschaft der Wissenschaften, mathematisch-physikalische Klasse*, Göttingen, pp. 198-219, 1897 (also in: H. Minkowski, *Gesammelte Abhandlungen*, Leipzig 1911).
- [19] A. V. Pogorelov, *Differential Geometry*, Noordhoff-Groningen, The Netherlands, 1956.
- [20] L. A. Lysternik, *Convex Figures and Polyhedra*, Dover, N. Y., 1963.
- [21] J. J. Little, "An Iterative Method for Reconstructing Convex Polyhedra from Extended Gaussian Images," *Proc. National Conf. on Artificial Intelligence*, August 1983, Washington, D. C., pp. 247-254.
- [22] D. Hilbert and S. Cohn-Vossen, *Geometry and the Imagination*, Chelsea Publishing, N. Y., 1952.
- [23] B. O'Neill, *Elementary Differential Geometry*, Academic Press, 1966.
- [24] G. A. Korn and T. M. Korn, *Mathematical Handbook—for scientists and engineers*, McGraw-Hill, 1968.
- [25] M. P. do Carmo, *Differential Geometry of Curves and Surfaces* Prentice Hall, Englewood Cliffs, 1976.
- [26] A. Alexandroff, "Existence and Uniqueness of a Convex Surface with a Given Integral Curvature," *Comptes Rendus (Doklady) de l'Académie des Sciences de l'URSS*, vol. 35, no. 5, pp. 131-134, 1942.
- [27] M. J. Wenninger, *Spherical Models*, Cambridge University Press, 1979.
- [28] L. Fejes Toth, *Regular Figures*, Pergamon Press, N. Y., 1964.
- [29] M. J. Wenninger, *Polyhedron Models*, Cambridge University Press, 1971.
- [30] H. S. M. Coxeter, *Regular Polytopes*, Dover Publications, N. Y., 1973.
- [31] H. Kenner, *Geodesic Math—and how to use it*, University of California Press, 1976.
- [32] P. Pearce and S. Pearce, *Polyhedra Primer*, Van Nostrand Reinhold, N. Y., 1978.
- [33] A. Pugh, *Polyhedra—a visual approach*, University of California Press, 1976.
- [34] C. M. Brown, "Representing the orientation of dendritic fields with geodesic tessellations," Internal Report, TR-13, Computer Science Department, University of Rochester, 1977.
- [35] B. K. P. Horn, "The Curve of Least Energy," *ACM Transactions on Mathematical Software*, vol. 9, no. 4, pp. 441-460, December 1983.

Symmetry Evaluators

S.A. Friedberg and C.M. Brown
Computer Science Department
University of Rochester
Rochester, NY 14627

Abstract

It has been known for some time that a bilaterally symmetric figure in an arbitrarily oriented plane P, viewed under orthography, yields a skew symmetric figure whose axes of symmetry and skew constrain the orientation of P to a one-parameter subspace of the spherical space of orientation vectors.

In recent work aimed at finding skew and symmetry axes of shapes, we used the measured moments of input skewed figures to constrain their symmetry and skew axes to a one-parameter subspace of the toroidal space of axis angle pairs. This subspace may be searched for those axis pairs yielding maximum symmetry in the original figure.

Here we briefly review the original constraint and then consider several symmetry evaluators for use in axis-finding. The most reliable proves to be one based on the mass in radial segments of the figure.

Key words: symmetry, measure, skewed symmetry, symmetry constraint, discrete image shape.

1. Skewed Symmetry and The Axis Constraint

A symmetric shape on a plane P viewed orthographically from an arbitrary viewpoint exhibits skewed symmetry. In the image, let the angle of the axis of symmetry from the horizontal be α , the angle of the axis of skew from the horizontal be γ . The skew of the figure is $\gamma - \alpha = \beta$.

Any two of α , β , γ constrain the plane P to a hyperbola of orientations in gradient space [Kanade, 1979; Kender, 1978; Stevens, 1979]. A method for finding α , β , and γ from image input was given in [Friedberg, 1984; Friedberg and Brown, 1984]. Briefly, α and γ may be expressed in terms of the second moments of the input shape. The resulting equations constraint α and γ to a closed curve on the toroidal (α, γ) space which is guaranteed to contain the correct α and γ values, of which there may be several. This locus must then be searched for (α, γ) values that are "correct", i.e. that explain the input shape as a symmetric original shape viewed under orthography. For example, any image triangle has three such (α, γ) pairs.

Here, as in previous work, we ignore the problem of searching the locus -- in fact, in our work we evaluate all points on it. The main content of the following sections is the various evaluators for (α, γ) pairs in the search. Each of these evaluators may be simply regarded as a measure of (non-skewed bilateral) symmetry. To evaluate an (α, γ) pair, the skewed figure is (implicitly or explicitly) "deprojected" through the (α, γ) skew, that is applying the inverse skew, and the symmetry of the resulting figure (across the x axis) is evaluated (Fig. 1).

Briefly, to recapitulate the mathematics of the constraint on (α, γ) , first note that the 3-D orthographic projection is equivalent to a 2-D skew of original points x_0 to yield points x_i

$$x_i = x_0 S, \text{ where}$$

$$S = \begin{bmatrix} \cos \alpha & \sin \alpha \\ \cos \alpha \cot \beta - \sin \alpha & \sin \alpha \cot \beta + \cos \alpha \end{bmatrix}$$

The original figure has a second-moment matrix

$$M = \begin{bmatrix} m_{20} & m_{11} \\ m_{11} & m_{02} \end{bmatrix}$$

in which $m_{11} = 0$.

The moment matrix for the skewed image shape is $M' = S^T M S$, and this implies the constraint C:

$$\alpha = \tan^{-1}((m'_{11} \tan \alpha - m'_{02}) / (m'_{20} \tan \gamma - m'_{11}))$$

$$\gamma = \tan^{-1}((m'_{02} \cot \alpha - m'_{11}) / (m'_{11} \cot \alpha - m'_{20}))$$

where the m'_{ij} 's are measured in the image. Constraint C implies α and γ lie on a 1-D locus in (α, γ) space.

2. Practicalities

The evaluator research is concerned with noisy digital images, not continuous mathematics. Quantization and noise effect have considerable effect on mathematically perfect evaluators. Further, it has so far been concerned with reliable symmetry detectors, not those that mirror human performance. We find that evaluators based on boundary features, while possibly relevant for humans, do not, in the simple form we implemented, perform reliably or predictably.

3. Lateral Symmetry Evaluators

All the measures discussed in this section require a significant amount of computation to evaluate. This is due to the need to invert the candidate (α, γ) transformation on a point by point basis. E_{strict} (Section 3.1) is worst in this regard since it requires transforming two points completely for each point in the figure. E_{3m} (Section 3.3) is best since it requires determining only the ordinate of each point in the figure. All the measures in this group require processing an entire figure to judge its degree of symmetry. This precludes time-saving techniques like abandoning a candidate axis pair if the evaluation passes some heuristic threshold. Experimental results with these evaluators are given in Section 7.

3.1 Strict Symmetry

An obvious evaluator is based on the definition of symmetry. Strict symmetry requires that $D(x_0, y_0) = D(x_0, -y_0)$, where D is density, intensity, texture or some other feature of interest. Let our measure of strict symmetry be defined as:

$$E_{\text{strict}} = \sum |D(x_0, y_0) - D(x_0, -y_0)|$$

Deprojecting through the best (α, γ) minimizes E_{strict} . Our definition of skewed symmetry for a figure is equivalent to finding $E_{\text{strict}} = 0$ when measuring original point locations in the coordinates implied by some (α, γ) pair. Clearly this measure is perfect in the continuous realm. Unfortunately, under the discrete representation E_{strict} performs poorly. The most significant problem is the dislocation of fine detail. E_{strict} is based on one-to-one point pairings and is very sensitive to dislocation, especially of high-frequency features of interest such as boundaries.

3.2 Product of Inertia

More successful than the one-to-one point pairing are some necessary but insufficient conditions for symmetry. Our solution for the constraint C was based on the product of inertia, moment m_{11} , being zero for symmetric figures. Let us define the product of inertia measure as:

$$E_{\text{poi}} = |m_{11}|$$

where m_{11} is the moment measured in the original figure, not the image. In practical terms, we first deproject by applying the inverse of the candidate (α, γ) skewing and then measure m_{11} . Probably because of its global nature, this measure is less sensitive to dislocation than E_{strict} (a property it shares with other moment measures).

It appears E_{poi} lacks discriminatory power due to the nature of constraint C itself. Since we exploit the property $E_{\text{poi}} = 0$ for symmetric figures in the solution for C , the (α, γ) candidate pairs in the locus for a given

figure are just those that transform some symmetric figure into one with the moments measured in the image. The observed range of values for E_{poi} is unsatisfactorily small and close to the minimum value of zero.

3.3 Third Moment

A measure of a symmetry (used in statistics to measure skewness) is the third moment in y , m_{03} . Define:

$$E_{3m} = |m_{03}|$$

Again, we measure this moment in the original figure, not the image. The measure is minimized by a symmetric figure.

This measure exhibits at least one major weakness: E_{3m} is minimized by antisymmetries as well as symmetries. This is most noticeable when evaluating parallelograms, which are judged to have an infinite number of equally good (α, γ) when there are actually only four. For many general figures, E_{3m} has proven quite effective in discriminating axes of skewed symmetry.

4. Radial Section Symmetry

This technique is based on the polar coordinates of points in the figure rather than their Cartesian coordinates. It has been the most effective evaluator in our experiments (Section 7). Divide a figure into n sectors by drawing n equally spaced rays with their endpoints at the center of the figure, aligning the positive ray of the x axis with one of these sector boundary rays. Number the sectors 1 to n clockwise from the sector whose counter-clockwise border is aligned with the x axis.

Let the area of the figure falling within sector i be A_i . For a symmetric figure A_i and A_{n-i+1} will be equal for $1 < i < n$. Define the measure of sector symmetry as:

$$E_{ss} = \sum |A_i - A_{n-i+1}|, \quad 1 < i < n/2$$

E_{ss} is minimized for a symmetric figure.

A_i may be quickly calculated. Rather than transform the points in the image figure and place them in the untransformed sectors, we may place the untransformed points in transformed versions of the sectors i . We exploit the fact that any angle θ_0 (measured relative to the x axis) in the original figure is skewed to an angle θ_i in the image figure by the relation:

$$\theta_i = \tan^{-1}(\tan \theta_0 + \cot(\gamma - \alpha)) + \alpha$$

The transformed sector boundaries can thus be easily calculated. We only need to calculate the θ_i for each point in the image figure once, since they are measurable in the image and do not depend on the candidate skew transformation.

A suggested implementation is to calculate a sorted list of the θ_i once. For each candidate (α, γ) pair generate the transformed sector boundaries. Find the sector in which the least θ_i falls. Test each θ_i against the upper boundary for the current sector. Increment the area for the current section if θ_i is less than the sector boundary. Advance to the next sector if not.

Using 32 sectors, evaluations of (α, γ) pairs as close as five degrees total difference in α and γ vary by a factor of eight or more when one of the pairs is a solution. The result is a high degree of discrimination among (α, γ) transformations. Adaptations of E_{ss} that substitute polar moments over sectors rather than areas of sectors could be developed. Since such moments would essentially measure powers of the points' distance from the origin, which is invariant under skew, they can be readily calculated from measurements in the image. We have not found the need for such adaptations, even for highly distorted image figures or image figures derived from unconnected originals.

5. Compactness

A widely-used shape number is compactness, defined as area/perimeter². Define:

$$E_{com} = A/P^2$$

which takes on the maximum value of $1/4\pi$ for a circle. Brady and Yuille use compactness to determine three-dimensional orientation from two-dimensional contour. They show that compactness is maximized if a skew symmetric figure is "unskewed" to one of its original symmetric shapes [Brady and Yuille, 1983]. For a somewhat related measure, ellipticity, see [Proffitt, 1982].

We have implemented a version of the measure for our discrete representation. We found it lacking in discriminatory power (Section 7). Although area and perimeter are elegantly calculated in the world of continuous figures, they are not well behaved in the discrete realm. Perimeter may be especially troublesome to compute accurately in comparison with other measures. Rosenfeld reports experiments where compactness of large digitized circles was computed using two different measures of perimeter. For one measure, the discrete compactness was maximized at approximately .98; for the other measure, the maximum was approximately .73 [Rosenfeld, 1974]. Compare these figures with the expected figure of about .79. For additional discussion of computing compactness over discrete figures see [Sankar and Krishnamurthy, 1978] and [Kulpa, 1979].

Our version of E_{com} is based on the A_i defined for use with E_{ss} . Approximate the radius for sector i by $r_i = \sqrt{A_i/\pi}$ and the perimeter for sector i by $P_i = 2\pi r_i/n$. From this, approximate the total perimeter by $P = \sum P_i + \sum |r_i - r_{i+1}|$, and calculate the ratio A/P^2 .

The standard technique of calculating area and perimeter length concurrently while following a chain code representation of the figure's perimeter may be adapted to the purpose [Freeman, 1974]. This technique is a discrete analogue of the continuous integration for area by Stokes theorem. There are two ways to approach an adaptation for figures under various (α, γ) transformations. The first is to take the chain code derived from the image figure, make a table of transformations from angles and distances in the image figure to angles and distances in the candidate transformed figure and use these transformed measures in the usual calculation. The second is to transform the figure, define the new perimeter and the corresponding chain code and proceed as usual. The first approach introduces less quantization noise and has the advantage of being much cheaper computationally.

E_{com} works best with connected figures but can be used with unconnected figures once the separate components have been identified. It has the computational advantage that the amount of perimeter information required can be substantially less than the information required by other measures to evaluate symmetry. This essential dependence on the perimeter means initial determination of the perimeter is critical and subject to some of the same resolution issues that affect tangent and curvature computation.

E_{com} inherently resolves shear ambiguity in favor of the most compact shape. Other measures of symmetry respond equally favorably to all shear-ambiguous solutions. E_{com} produces local maxima for the (multiple) solutions, but it may not be possible to determine the number of actual solutions, especially since some solutions may be much less compact than others of equal symmetry.

6. Boundary Features and Hough Transform Evaluators

The Hough transform is a technique with a wide range of applications. Typically, a set of operators is applied uniformly over an input parameter space. The results of evaluation at each point in the image space are consistent with certain values in the space of parameters to be determined. This has been likened to "voting" for the appropriate parameters. In this way, local information in the image space is transformed to a global solution in the parameter space. Experimental results for the techniques below appear in Section 7.

6.1 Boundary Pair Matching

Take two points $(x1_i, y1_i)$ and $(x2_i, y2_i)$ on the perimeter of a figure defining a chord across the figure or a lobe of it. Each such pair of boundary points contributes one vote in the parameter space. (α, γ) :

$$\alpha = \tan^{-1}((y1_i - y2_i) / (x1_i - x2_i))$$

$$\gamma = \tan^{-1}((y1_i + y2_i) / (x1_i + x2_i))$$

That is, make the assumption that this chord is parallel to the axis of skew and the midpoint of the chord lies on the axis of symmetry. Symmetric figures maximize the contributions for (α, γ) pairs corresponding to axes of skewed symmetry.

Why is this a plausible evaluation? We distinguish illegal, legal and actual solutions. Illegal solutions are (α, γ) pairs that do not fall on the locus of constraint C. Legal solutions are pairs that do fall on the locus. Actual solutions must be legal solutions and they correspond to the actual skewed symmetries in the figure. If the two points are, in fact, symmetrically corresponding points in some legal solution, the chord does have the desired properties. If the two points are not symmetric points in a legal solution, the chord will contribute to an illegal (α, γ) pair in the parameter space.

We now need to determine that the voting process isolates the actual solutions among the legal solutions. Let the length of the projection of a figure onto the symmetry axis for some (α, γ) pair be L. Assume the figure is convex and connected. If (α, γ) is an actual solution, the evaluation of that pair will theoretically receive L votes. This corresponds to the number of chords centered on the axis of symmetry and parallel to the axis of skew that can be drawn across the figure. A non-solution will receive less than L votes, else it would be an actual solution.

Robust behavior is generally provided by the (sparse) distribution of votes over the parameter space, which tends to avoid clustering around legal but non-actual solutions. If we measure the "projection" of a non-convex perimeter onto the symmetry axis as $L = \sum (\Delta s \text{Pos}(\Delta x / \Delta s))$ where s is arc length along the perimeter and $(x > 0) \rightarrow \text{Pos}(x) = x$, otherwise $\text{Pos}(x) = 0$, then again an actual solution theoretically receives L votes.

So long as we compensate for changing L with changing (α, γ) , actual solutions can be discriminated from non-solutions. In the absence of such compensation, this Hough transform generally demonstrates an acceptable degree of discrimination between solutions and "close" non-solutions and a moderately better degree of discrimination between solutions and "background." The need to compensate evaluations differently for each candidate transformation poses a practical problem for Hough transform methods.

6.2 Tangent Pair Matching

A variant of the boundary pair matching idea uses the following observation. Let τ_1 and τ_2 be tangents at points $[x_1, y_1]$ and $[x_2, y_2]$ in a symmetric figure. Then the intersection of τ_1 and τ_2 :

- 1) lies on the axis of symmetry, and
- 2) is invariant under skew (but not rotation).

Let γ be calculated as before and let

$$\Delta x = x_2 - x_1$$

$$\Delta y = y_2 - y_1$$

$$\alpha = \tan^{-1}((\tan \theta_1 y_2 - \tan \theta_2 y_1 - \tan \theta_2 \Delta x) / (\Delta y - \tan \theta_2 x_2 + \tan \theta_1 x_1))$$

where θ_1 and θ_2 are the angles of the tangents at $[x_1, y_1]$ and $[x_2, y_2]$ respectively. The value for α is determined by the angle between the intersection of the tangents and the centroid.

6.3 Strict Lateral Hough

This is an adaptation of strict lateral symmetry. Each pair of points in the figure, not just the points on the perimeter, contribute to the overall result. Otherwise, α and γ are calculated in the same fashion as for boundary pair matching.

6.4 Discussion

These techniques are significantly more expensive than the others discussed. The two perimeter-based versions run in time proportional to the square of the length of the perimeter, rather worse than the other perimeter-based techniques. The strict lateral technique runs in time proportional to the square of the area, again worse than the other area-based techniques.

An additional disadvantage is that to accumulate partial results over the entire parameter space. Doubling angular resolution, axis solutions quadruples the size of the parameter space. It is possible in some sense to invert the technique, as used here by evaluating transformed figures for orthogonal axes rather than directly evaluating image figures for axes of skewed symmetry, but this removes one of the potential benefits of using a Hough transform. Fortunately, the constraint C allows us to discard any results that do not contribute to legal solutions. Doing so saves considerable space at the expense of minor additional complexity.

Both perimeter based techniques have a strong bias for axes of symmetry that align with parallel lines in the figure or with the axis of elongation. The preference for parallel lines is due to the heavy contributions to precisely the same α . The preference for the axis of elongation is due to the relationship between the projection onto the symmetry axis and the number of votes that can be accumulated, as noted above. This is most noticeable when evaluating irregular parallelograms. One symmetry axis is clearly preferred over the other three solutions.

The tangent pair matching technique demonstrates one advantage over the boundary pair matching technique. The undesirable preference for parallel lines can be adjusted by weighting contributions by some function of the difference between the two tangents. It has the disadvantages of additional computational cost and a requirement for higher resolution representation for equivalent accuracy results.

Hough techniques have two major redeeming values in this application. First, the perimeter techniques are robust enough to deal with obscuration since no portion of the perimeter is critical to the overall result. Second, the techniques evaluate all candidate (α, γ) pairs in one pass over the image rather than requiring a separate evaluation of each candidate pair. When the angular resolution desired is high, this may swing the overall running time for evaluation of skew symmetry to favor the Hough transform. Clearly we lose most of this feature if we are forced to process each possible transformation separately. An additional potential of these evaluators over others is the inherent parallelism of Hough transforms.

7. Experiments

Evaluation procedures were implemented for E_{struct} , E_{per} , E_{3m} , E_{ss} , E_{com} , and the three Hough transform techniques. The performance of each implemented evaluator was tested on at least 10 different figures in judging (subjectively) their relative merits.

Calculation of the locus of constraint C was performed once, and as many as four evaluators could be invoked on each candidate transformation. The Hough transforms were implemented separately, and the constraint C was used to "filter" the results of applying each transform. Graphic displays of the locus of constraint C and intensity plots of the evaluators were implemented.

Figure 2 serves as a guide to Figures 3, 4, and 5, which show the eight evaluators applied to three input shapes.

8. Conclusions and Future Work

We present and test eight evaluators of symmetry, some from the literature and some new. The sector symmetry evaluator provides the best combination of accuracy and speed. This paper is a condensation of a technical report [Friedberg, 1984], in which eleven symmetry evaluators are presented and analyzed at greater length.

Analysis of the sensitivity of various evaluators to different kinds and degrees of asymmetry can provide a more objective method of choosing among the (many) alternatives. The evaluators discussed here and others encountered in the literature should prove amenable to such analysis. Particularly interesting forms of asymmetry are those introduced when processing real images, of course.

If there are constraints on the original shape of the figure being processed (e.g., figures are known to be portions of automotive connecting rods) or if there are constraints on the axes of skewed symmetry (e.g., in aerial photography with known relation between camera and ground coordinates) additional analytic constraints may be developed.

Skew transformations significantly alter some forms of texture. Consider radial lines as a texture pattern. In an unskewed figure, these lines are of uniform density around the figure. In a skewed figure, the density of these lines varies with position and the axes of skewed symmetry can, in fact, be recovered from the variations in texture density. A generalization of this behavior and its exploitation would provide a useful tool in processing obscured images or in further constraining the axes of skewed symmetry for visible ones.

If we adopt an imaging model with perspective projection, we can no longer use constraint C on shear ambiguity or Kanade's constraint on gradient ambiguity. Many tasks in robotics and "near" vision must process images produced under perspective projection. Generalizing either constraint would be useful.

Finally, we make no claims that the symmetry evaluators presented here have any relation to analogous processes in the human visual system [Friedberg and Brown, 1984; Schaefer, 1984].

Bibliography

- Ballard, D. H. and Brown, C. M., *Computer Vision*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982.
- Brady, M. and Yuille, A., "An Extremum Principle for Shape from Contour," AI Memo 711, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, January 1983.
- Brucks, M. L., personal communication, October 1983.
- Davis, L. S., "Understanding Shape: II. Symmetry," *IEEE Trans. Sys. Man, Cybernet.*, SMC-7, 3, 204-212, 1977.
- Ellis, T. J., Proffitt, D., Rosen, D. and Rutkowski, W., "Measurement of the Lengths of Digitized Curved Lines," *Computer Graphics and Image Processing*, 10, (4), 333-34, 1979.
- Freeman, H., "Computer Processing of Line-Drawing Images," *Computing Surveys*, 6, (1), 57-97, 1974.
- Friedberg, S. A., "Finding Axes of Skewed Symmetry," TR127, Computer Science Dept., University of Rochester, November 1983.
- Friedberg, S. A. and C. M. Brown., "Finding Axes of Skewed Symmetry," *Proceedings, 7th Int'l. Conf. on Pattern Recognition*, Montreal, Canada, August 1984.
- Friedberg, S. A., "Symmetry Evaluators," TR134, Computer Science Dept., University of Rochester, September 1984.
- Hoffman, D. D., "Representing Shapes for Visual Recognition," Ph.D. thesis, Dept. of Psychology, Massachusetts Institute of Technology, 1983.

Kanade, T., "Recovery of the Three-Dimensional Shape of an Object from a Single View," CMU-CS-79-153, Computer Science Department, Carnegie-Mellon University, October 1979.

Kender, J. R., "Shape from Texture: A Brief Overview and a New Aggregate Transform," *Proceedings, DARPA Image Understanding Workshop*, Carnegie-Mellon University, 79-84, November 1978.

Kulpa, Z., "Area and Perimeter Measurement of Blobs in Discrete Binary Pictures," *Computer Graphics and Image Processing*, 6 (5), 434-451, 1977.

Kulpa, Z., "On the Properties of Discrete Circles, Rings and Disks," *Computer Graphics and Image Processing*, 10 (4), 348-365, 1979.

Newman, W. M. and Sproull, R. F., *Principles of Interactive Computer Graphics*, 2nd Ed., McGraw-Hill, New York, 1979.

Parui, S. K. and Majumder, D. D., "Symmetry Analysis by Computer," *Pattern Recognition*, 16 (1), 63-76, 1983.

Pavlidis, T., *Structural Pattern Recognition*, Chapters 7, 8 and 9, Springer-Verlag, Berlin, 1977.

Proffitt, D., "The Measurement of Circularity and Ellipticity on a Digital Grid," *Pattern Recognition*, 15 (5), 383-387, 1982.

Proffitt, D. and Rosen, D., "Metrification Errors and Coding Efficiency of Chain-Encoding Schemes for the Representation of Lines and Edges," *Computer Graphics and Image Processing*, 10 (4), 318-332, 1979.

Requicha, A. A. G., "Representations of Rigid Solid Objects," *Computing Surveys*, 12 (4), 437-464, 1980.

Rosen, D., "On the Areas and Boundaries of Quantized Objects," *Computer Graphics and Image Processing*, 13 (1), 94-98, 1980.

Rosenfeld, A., "Compact Figures in Digital Pictures," *IEEE Trans. Sys. Man, Cybernet.*, SMC-4, 2, 221-223, 1974.

Sankar, S. V. and Krishnamurthy, E. V., "On the Compactness of Subsets of Digital Pictures," *Computer Graphics and Image Processing*, 8 (1), 136-143, 1978.

Schaefer, B. A., "A Model of Human Symmetry Detection," *Proceedings, 7th Int'l. Conf. on Pattern Recognition*, Montreal, Canada, 509-511, August 1984.

Stevens, K. A., "Representing and Analyzing Surface Orientation" in *Artificial Intelligence: An MIT Perspective*, Vol. 2, P. H. Winston and R. H. Brown (eds.), MIT Press, Cambridge, Massachusetts, 1979.

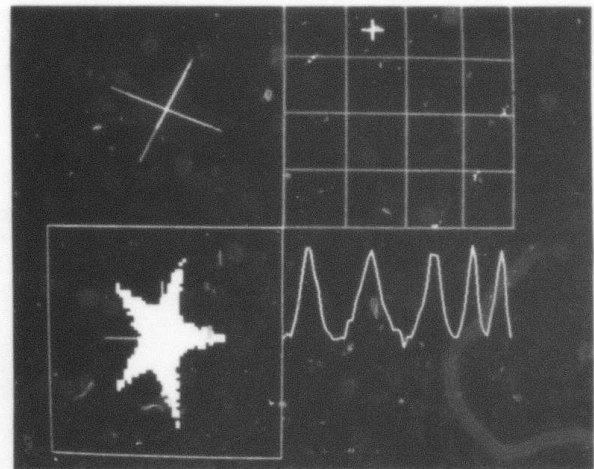


Figure 1. An input image (upper left), the locus of (α, γ) weighted in intensity by the evaluator output (upper right), the evaluator output displayed as ordinate with α as abscissa (lower right), and the symmetric result of deprojecting the input figure through the (α, γ) skew corresponding to one of the five maxima in evaluator output (lower left).

Input Shape	(α, γ) Locus
	Radial Sector Symmetry
Whole Area Hough Transform	(α, γ) Locus
Boundary Tangent Hough Transform	Boundary Location Hough Transform
Strict Lateral Symmetry	Compactness
Product of Inertia m_{11}	Third Moment m_{03}

Figure 2. Guide to Figures 3, 4, 5. In each case the evaluator outputs are highlighted at the correct (α, γ) .

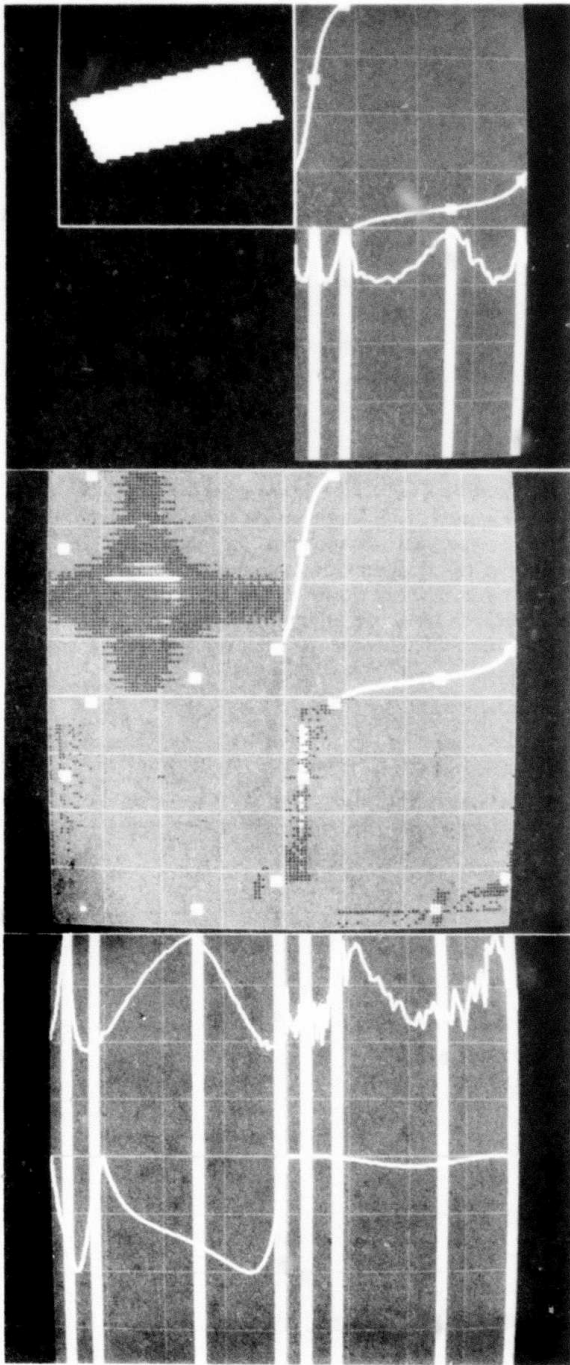


Figure 3: The radial sector evaluator performs best on the parallelogram, with the strict lateral evaluator finding the rectangular solutions and the product of inertia finding the rhomboidal solutions. Of special interest is the third moment, which is uniformly good since it is maximized under both symmetry and antisymmetry. The Hough Transform measures do not perform well, with the boundary location scheme working best.

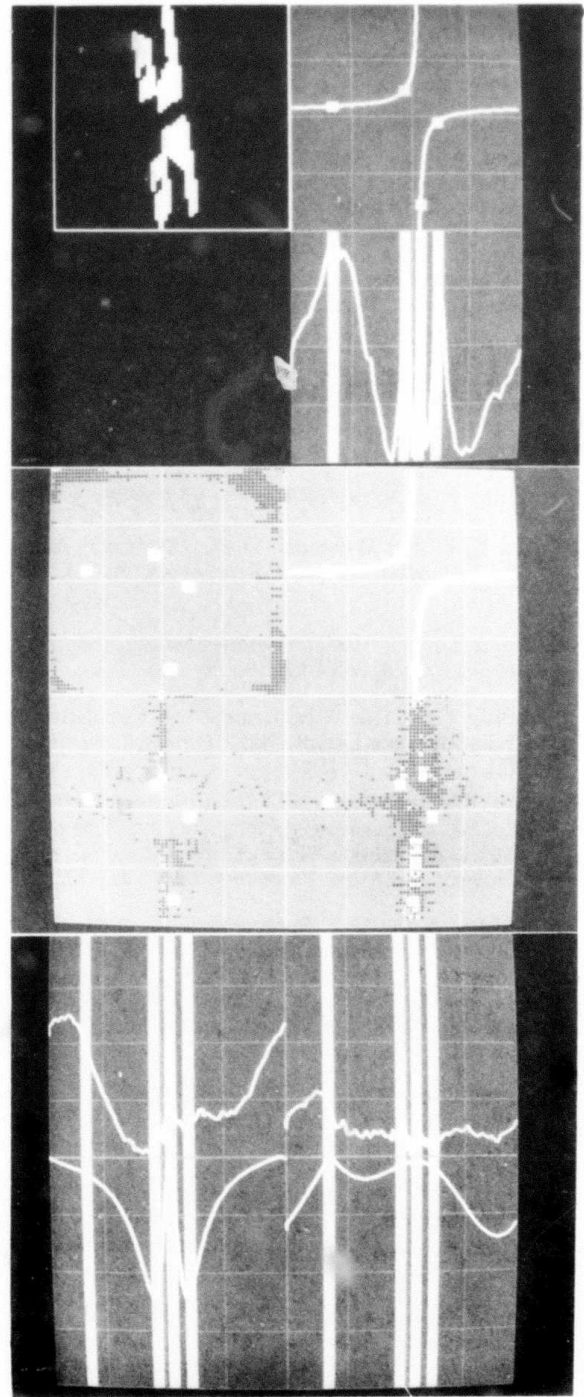


Figure 4: For a complex "thunderbird" shape, the radial sector symmetry again performs best, with all other measures performing disastrously except for compactness, which finds two solutions.

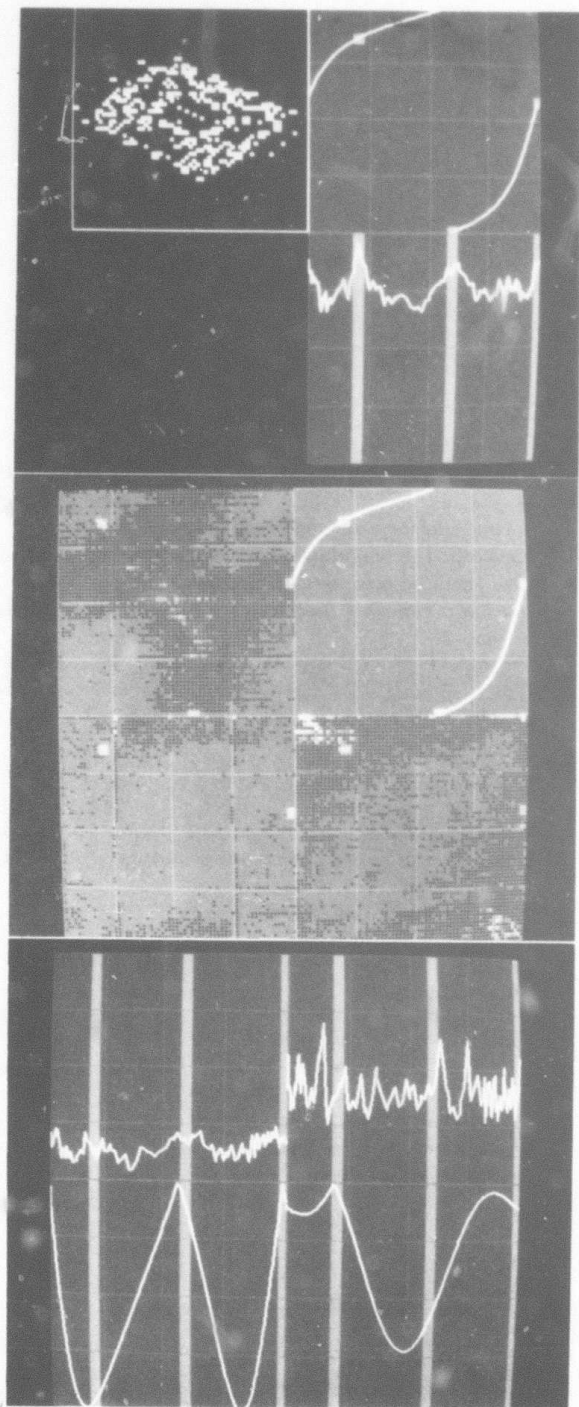


Figure 5: A "shape" made by reflecting a random cloud of points confuses all strict symmetry and feature-based schemes. Again, the lateral sector symmetry performs best. Moment evaluators respond strongly to some solutions.

GRAPHICS AND PREDICTION FROM MODELS

Richard Scott

Computer Science Department
Stanford University, Stanford, California 94305

Abstract

The first part of this paper describes algorithms and complexity arguments for a near optimal hidden surface scheme. It starts with a viewpoint and a scene composed by unions, intersections, negatives of parametrized volumes. Then limbs are found on the surfaces to given accuracy, and mutually consistent T-junctions and cusps are found in their projections to the image plane. The surfaces and image planes are represented by quad-trees. From these, the topology of the projection is extracted, represented by a "tooth-pick" structure which orders the surface regions, that project into the same area of the image. The foremost regions can be displayed for hidden surface graphics. The projection topology is invariant over a range of viewpoints and models. The second part of the paper examines the projection topology for changing viewpoint and surface shape, and discusses how it might be predicted and represented.

Introduction.

There are two parts to this paper. The first describes an optimal (and mainly implemented) scheme for a variation of the hidden surface problem. This is graphics whose goal is not just to display the image, but to know qualitatively what it is displaying. Its output represents the topology of the projection mapping in a "tooth-pick" graph structure, which aligns regions of surface and image, and at whose nodes the quantitative details are stored.

Graphics can be thought of as prediction with fixed viewpoint and surface shape. The projection topology structure is invariant to small changes in viewpoint and surface shape. The second part of the paper discusses general prediction by considering how the projection topology can change as viewpoint and surface shape vary.

So three different mappings are analysed, with decreasing success,

$P1 : \text{surface} \rightarrow \text{image}$

$P2 : \text{viewpoint} \rightarrow \text{topology-of-} P1,$

$P3 : \text{surface-shape} \rightarrow \text{topology-of-} P2.$

The method of attack in all cases is,

1. What are the singularities, where a small change in the left hand side produces a large change in the right hand side? How can they be found and represented?
2. How can the singularities, and the continuous spaces between them be organized and represented?
3. How can a complete description of the map be extracted from 1 and 2?

In the case of P1, the method is justified by complexity arguments in section 3. The rest of the paper is organized as follows:

(Introduction.)

Part One.

1. Geometry of the projection mapping. Define some terms.
2. Complexity arguments to derive the outline of the hidden surface algorithm.
- 3.1 The three algorithm steps.
- 3.2 Supporting algorithms.
- 4.1 Modelling outline.
- 4.2 More Modelling.

Part Two.

- 5.0 Overview.
- 5.1 Singularities for changing view.
- 5.2 Singularities for changing surface shape.
6. Prediction Structures.

Part One.

1. Geometry of the Projection Mapping.

Before describing the algorithms, a few geometrical terms are defined.

A *limb*, also called a *contour-generator* is a closed loop of points on the surface being looked at, where the line of sight is tangent to the surface, i.e., perpendicular to the surface normal.

A *contour* is the projection of a limb to the image plane.

A *T-junction* is a point in the image where two contours cross. There are two different limb points on the same line of sight, the one closer to the viewpoint occludes the other.

The limbs divide the surface into forward and backward facing regions compared to the eye, and are oriented so that the forward face lies on the left (when looking down on the surface from outside). Let F be the line of sight vector, and L the limb tangent vector. Both F and L lie in the surface tangent plane. If $F \vee L$ is parallel to N (the outward surface normal) then the forward face, on the left of L , is closer to the eye than the right side, giving an *outer limb point*.

If $F \vee L$ is anti-parallel to N , then the back face, on the right, is closer, an *inner limb point*.

If $F \vee L$ is zero there is a *cusp point*. F parallel to L means that the tangent plane intersects the surface along F , which is the definition an *asymptotic direction* in the surface.

So L is parallel to an asymptotic direction in a hyperbolic (saddle shaped) region of surface. This is another way of defining a cusp. There are two types of cusp, with L pointing towards or away from the eye.

Fig. 1 shows an oblique view of a torus.

Whitney in 1955 proved that limbs and cusps are the only possible singularities from a generic viewpoint.

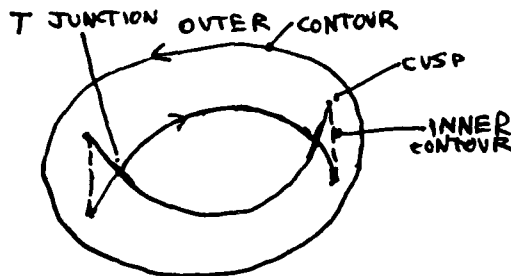


FIG. 1.

2. Hidden Surface Complexity Arguments.

These will show that an optimal algorithm must find the singularities; limbs, Ts and cusps.

If e is the maximum allowed error in the image, l is the total length of the limbs, c the total curvature of the limbs, then it has optimal complexity

$$O(n \log n), \text{ where } n \text{ has } O(\sqrt{lc/e}).$$

There is a sequence of refinements to the following first attempt algorithm.

1. Shoot out a conical bunch of rays from the eye. Calculate where they intersect the surfaces of the object being viewed, and order the intersections along each ray by distance from the eye.

Algorithm 1 is inefficient because neighbouring rays (image points) have the same ordering of surfaces along them, except when an intervening ray goes through a limb i.e., is tangent to the surface there. So the order of surface regions from the eye along an arbitrary ray can be recovered from the ordering along the subset of limb rays.

2. Find the intersections of limb rays (tangent to some surface) with other surfaces in their paths.

This is inefficient because between neighbouring limb points the surface orderings can only change in two ways. When an intermediate contour ray passes through (i) a T-junction, (ii) a cusp.

3. First find the limb rays then find the subset of T junctions and cusp rays and the intersections with other surfaces there.

However, to find the complete surface orderings along rays through cusp and T points, it is not necessary to solve for the intersections and then order them. Instead, the difference in surface orderings are already known:

- (i) Over a limb, the forward and backward facing regions are inserted together at one position in the ordering. For an outer limb segment, the forward face comes before the back face, while for an inner segment, the back face comes first.
- (ii) Over a cusp, the forward and backward facing regions, adjacent in the order, swap places.
- (iii) Over a T, the two regions of the other T branch insert themselves.

This set of differences can be solved to give complete surface orderings for each image region, by propagating partial orderings over the image.

4. Find the singular rays at limbs, T junctions and cusps, then solve for the unique, complete surface orderings in each image region, that satisfy the ordering differences at singular rays.

This is the optimal hidden surface algorithm. Still unspecified are the details of

- (i) how the limbs, cusps and T junctions are to be found,
- (ii) exactly what surface regions are being ordered, (and how to ensure that the Ts and cusps of each region, solved locally to within an error bound, are globally consistent.)
- (iii) and how to propagate up the orderings.

These are described in the next section, but as far as complexity is concerned, (ii) and (iii) are both $O(\text{number of image areas between contours})$ which is a measure of the scene complexity, of smaller cost than (i).

Let e be the maximum allowed error distance between the approximate and exact contours in the image, l be the total length of the limbs, c be the total curvature of the limbs, and n be the number of limb rays solved for.

The average step length between adjacent rays is l/n , the average step angle is c/n , and the interpolation error is $O(\text{step-length} \times \text{step-angle})$ so, $O(lc/n^2) \leq O(e)$, and n must be $O(\sqrt{lc/e})$.

Meanwhile, the cost of solving iteratively for one limb point, is $O(\log(\text{initial-error}/\text{final-error}))$. The final-error is $O(e)$, and the initial-error is $O(\text{step-length} \times \text{step-angle})$, which makes the cost $O(\log(lc/en^2)) = O(1)$.

This means that the cost of solving for all n contour rays is $O(n)$. The cost of finding each cusp is $O(1)$; just see if the limb to ray angle has changed sign. To find T junctions, the n contour segments have to be tested for intersection with each other, costing $O(n \log n)$. (Assuming that no topological information is being used to direct the search.)

Adding $O(n)$ and $O(n \log n)$, results in the complexity of an optimal hidden surface algorithm, $O(n \log n)$, where n is $O(\sqrt{lc}/\epsilon)$.

Actually the $O(n)$ limb solving dominates.

3.1. Three Steps of Hidden Surface Algorithm.

1. Split the whole surface up into maximal regions that project $(1-1)$ to the image. i.e., which do not twist around to hide themselves. These are sub-regions of the forward and back facing areas bounded by limbs.
2. Split the image plane up into areas that have a constant ordering of the surface regions that project onto them. Each area is bounded by a number of contour segments with Ts or cusps at the corners.
3. In each image area work out what the ordering actually is. There seems to be an optimal algorithm to do this which requires only the information provided by Steps 1 and 2.

For hidden surface graphics including transparency, the surface regions can be displayed in order, up to the first opaque one, for each area of the image.

Step One.

1. Step one first finds the limbs, which divide the surface up into alternating forward and backward facing regions called *faces* (w.r.t. the viewpoint). For planar surfaces tangent to the line of sight, the limb is taken along the front edge, so that the plane belongs to the back facing region. Each face is bounded by a set of different limbs called *limb sets*. Successive segments of the limbs are stored in quad-trees representing the coordinate patches of the surface. Regions of the quad-trees between limbs are filled to extract the limb sets.
2. The only way that a face can occlude itself is by having T junctions or cusps in its limb set. If there are none, then the face projects $(1-1)$ to the image.
3. Ts and cusps mean that the face needs to be split up further into $(1-1)$ sub-regions. There seems to be a decomposition algorithm that makes the splitting unique, (it pairs up Ts and cusps in the limb set), and at the same time ensures that the combination of Ts and cusps is consistent with a physically possible surface, (approximating the exact one). Having the Ts and cusps consistent for each face separately, $< >$ the whole image is consistent.

The result is a graph structure where the nodes are the $(1-1)$ surface regions and the links are adjacency along a limb, or projected limb indicated by the decomposition.

Step Two.

In order to search for T junctions in contour sets (the projection of limb sets), a representation of the image has

already been built up in Step 1. It is a quad-tree of part of the image plane. However since it is not yet being used for any distance measurements, it might be better to think of it as a small sphere surrounding the viewpoint, that the surfaces are being projected onto.

In this step T junctions between contours in different sets are found. There are no global consistency conditions for these Ts. The smallest undivided areas of the image are extracted from the quad-tree, to build a graph of the image where the nodes are areas and the links are the projected boundaries of the $(1-1)$ areas of Step one. The ends of the links are cusps or T junctions.

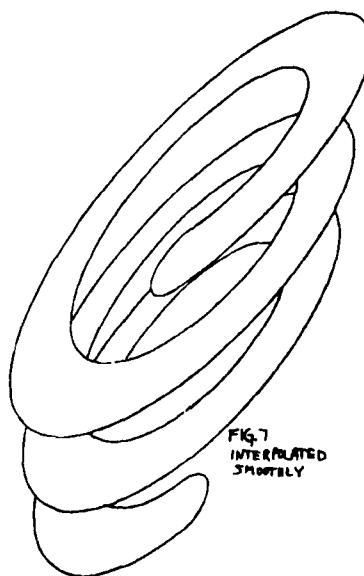
Step Three.

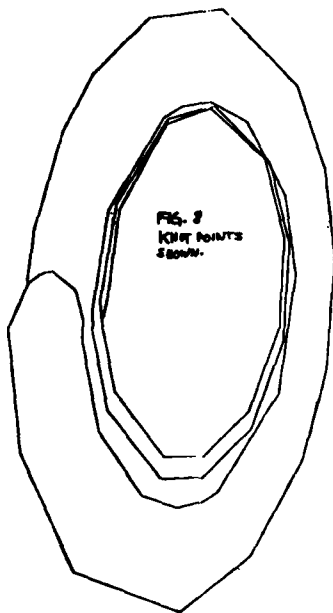
The last step quickly redistributes information already contained in the two graphs produced by steps 1 and 2, to get the ordering of surfaces in every image area.

It is important that the algorithm that does this can be proved to work in all cases; and that the ordering of surrounding contours is always resolved without having to intersect a new ray with both surfaces. For two disconnected objects, that share no T junctions, exactly one ray intersection is needed.

The graph of Step 2, says which $(1-1)$ regions of surface project over any image point. A sketch of the proof that they can be ordered pairwise, is given in 3.2.F. The details of the ordering algorithm have not yet been worked out.

These 3 steps outline the hidden surface scheme. The next sections give some more details. Some previous versions of these algorithms that found limbs, Ts and cusps without organizing the surface regions, have been implemented e.g., the two views of a helix, figs. 7,8.





3.2. Supporting Algorithms for Steps 1 and 2.

- A. Finding and following the limbs.
- B. Searching for T junctions and cusps.
- C. Decomposing into physically possible (1-1) subregions, giving image consistency.
- D. Refining T junctions.
- E. Image accuracy.
- F. Pairwise Ordering.

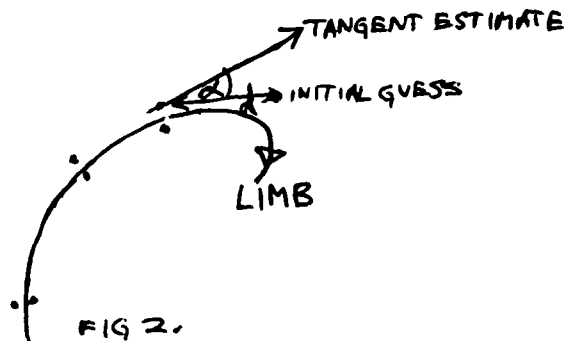
A. Finding the Limbs.

1. Search for a point on a new limb.

This is not a time consuming step; the algorithm steps along a fixed parameter path, and works out whether successive surface normals point towards or away from the eye. When they point oppositely, a limb passes between them. Has that limb already been solved? If so continue searching, otherwise give the two spanning points to the following algorithm.

2. Follow the limb over the surface, point by point, until it loops up with itself. This is the most expensive part of the whole hidden surface scheme.

Each limb point is actually a pair of points which span the limb closely. Having solved a stretch of limb, at the end point we know the tangent direction and the previous step angles, (see fig. 2). So an initial guess to start a new iteration for the next limb point can be made by extrapolating the curve some distance. The distance is chosen to minimize the number of iterations per step length. If the step length is too long, the Newton-Raphson may not converge, if too short, unnecessary extra points are found. It is chosen to give the constant, optimal number of iterations (3 or 4) that gets a bounding pair with angles between their surface normals small enough to give an accurate estimate of limb tangent for the next extrapolation.



In other words, step length d , and step angle α , along the extrapolation curve are chosen so that $d \sin \alpha$ is constant, which makes the initial error and number of iterations roughly constant.

If u and v are the two surface parameters, then the tangent direction in parameter space is, $du = d(n \cdot f)/dv$, $dv = -d(n \cdot f)/du$. This and the previous limb point are used to fit a curve to the limb in parameter space, that has linearly changing curvature. The extrapolation curve assumes linearly changing curvature.

B. T-Junctions and Cusps.

To find cusps on limbs check for change in sign of the triple scalar product,

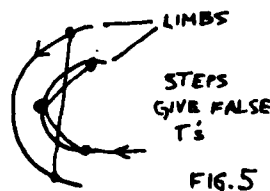
$$\text{Surface-normal} \cdot (\text{Line-of-sight} \vee \text{Limb-tangent})$$

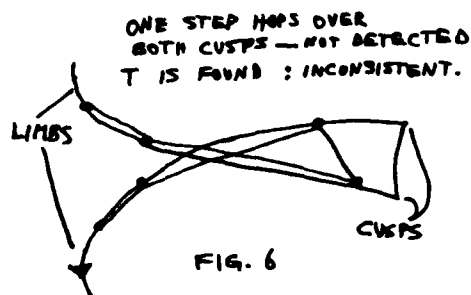
Finding T Junctions.

The image is represented by a quad tree, with clipping usually done around the borders of the unit square. When crossing contours are found in a quad square, they are initially refined until the ordering at the T is clear. Both ends of one segment must be closer to the eye than both ends of the other.

C. Decomposing the Faces.

The test to find cusps is local to a small patch of surface, and the T test is local to an area of the image. It is important that these local, approximate calculations can be combined into a globally consistent image. Fig. 5 shows two spurious T junctions. Fig. 6 shows the T junction of a fishtail detected without its cusps. As the accuracy constraints are relaxed, the resulting image should be the exact projection of a 3D volume that approximates the real one, degenerating into a sphere-like blob.





The decomposition algorithm has not been fully worked out, but here is a sketch of how it should go for each face.

Start with the surface graph for that face, as if it projected (1-1) with no Ts or cusps. This is a single node, with a link for each limb in its limb set. Find the obvious cusps. Use the image quad tree to detect Ts, and refine them to the desired image accuracy, using the (implemented) algorithm in the next section, D. If the desired accuracy is smaller than the distance between the false Ts, this will merge them away. Then link up the Ts and cusps around sub-regions which split the face and its graph node. Consistency conditions (paired Ts, cusps, etc.) are applied around the borders of the subregions, and extra Ts and cusps are added or removed to resolve inconsistencies. E.g., the two undetected cusps of Fig. 6 would be added.

D. Refining T junctions.

1. Two contour segments intersect each other. The longer contour is bisected (or a halfway interpolation), as an initial guess to iterate to a new limb solution point. If one of the two new halves crosses the shorter segment, this step can be repeated.

2. But what if it does not? Then one end of the short segment must be enclosed by the triangle formed by the long segment and its two halves. The contour at the enclosed end is extended until it leaves the triangle. If it crosses one of the halves, 1. can be resumed. But if it crosses the long segment, then it re-enters the stack of triangles formed by previous successful bisections. The contour continues being extended (step by step) through the stack, until it exits over a bisected side, and can return to 1. If this never happens, it finally cuts out of one of the original segments that formed a T in the quad-tree. In that case two quad-tree Ts can be merged away; they were just an artifact of approximate solution methods.

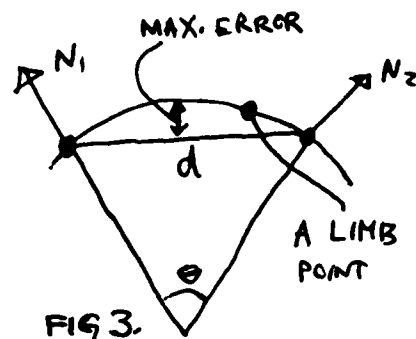
E. Guaranteeing Image Accuracy.

If $F1$ and $F2$ are the two bounds, in the frame of the eye, and are distance d apart, with θ the angle between their surface normals $N1$ and $N2$, then, $\text{sign}(N1 \cdot F1)$ is opposite to $\text{sign}(N2 \cdot F2)$.

The maximum possible image error is (see fig. 3)

$rk(1 - \cos \theta/2)$ where k is the ratio:
image-plane-distance/ $\min(|F1|, |F2|)$,
and r is $d/2 \sin \theta/2$.

$$\frac{dk \sin \theta/2}{2(1 + \cos \theta/2)} < e,$$



which reduces to, $d\sqrt{\frac{(1 - N1 \cdot N2)}{2}} < 2e/k$.

This is the condition for stopping the iteration that refines the bounds of a limb point.

For a projected step of length l in the image, with step angle α , the error in any interpolation scheme will be $O(l \sin \alpha)$. α is a measure of the inverse of radius-of-curvature, so the error is $O(l^2/\text{radius-of-curvature})$. There are two alternatives. If the image interpolating error needs to be bounded by fixed e , then the step length l between knot points should be $O(\sqrt{\text{radius-of-curvature}})$. However if the image interpolating just needs to look good, then the error should be proportional to step length. So $l \sin \alpha \propto l$, giving constant step angle, and step length $O(\text{radius-of-curvature})$.

F. Pairwise Ordering for Step 3.

This is the step that combines the information in the surface and image graphs, produced by Steps 1 and 2. Its output are the complete surface orderings in each image region, represented by lining up nodes of the surface and image graphs.

For each image region, we can get the set of surface regions that project over it. (By propagating around the inside of each contour set.) They can be ordered pairwise, hence completely.

Sketch of proof:

Take a pair of (1-1) surface regions A and B, that intersect in the image. There are two ways that they can be ordered immediately,

- (i) if they have a T junction between the image projections of their boundaries.
- (ii) if they share a boundary, are adjacent on the surface.

If neither of those works, find a connected path of regions from A to B over the surface of the object, i.e., through the graph from Step 1.

Step down the path, starting at A, checking each region against B by (i) and (ii), until one of them gives an ordering. It is the same ordering as between A and B themselves. We will eventually get an ordering because the last region in the path borders on B, and (ii) applies.

4.1 Modelling Outline, 3,2,1D.

3D Modelling.

Scene: Objects positioned in scene frame.
View: Scene transformed into eye frame.
Object: A connected object is given by:
Primitive volume, or
Deformed sub-object
(e.g. reflection, stretch, negative), or
union of sub-objects, or
intersection of sub-objects.
Primitive volume: Parametrized volume,
that is piecewise continuous
and differentiable.
e.g. Generalized cylinder,
half space of a plane.

This is theoretically equivalent to having the whole scene surface parametrized in a number of coordinate patches. Volume models are used because they are often more compact and easier to describe.

Examples of deformations:

One wing of an aircraft can be modelled as the reflection of the other. A tube will be the intersection of the outer cylinder with the negative of the inner one. The deformation of objects can be non-linear. E.g. looking at the scene through a distorting lens.

E.g. Definition of a screw, formed by cutting out the helical threads.

(Union (screw-head head-frame)

(Intersection (screw-body body-frame)

(Negative (screw-threads-helix body-frame))))

2D Modelling.

The image plane, and each coordinate patch of the surfaces have lines and regions represented by quad-trees. The lines are stored with step length roughly proportional to $\sqrt{\text{radius-of-curvature}}$, to give constant accuracy, as explained in section 3.2 E. The discrete version of this is, $\text{step-length} \propto \frac{1}{\sqrt{\sin(\text{step-angle})}}$, with a minimum length equal to the allowed image error. The quad squares are subdivided until the ends of a step are in different nodes. Example of use: Intersections of lines are found using the quad-trees.

1D Modelling.

Lines are represented by 2-way list of knot points, containing curvature information, and intersections with other lines.

Topological Modelling.

Projection mapping of viewed surfaces to image:

The topology is represented by a "tooth-pick" structure. A small number of tooth-picks, emanating from the eye, spear through surface regions of the scene and into regions of the image. They line up occluding patches of surface bounded by limb segments and their projections. This is invariant for a range of viewpoints and models.

4.2. More Modelling.

- A. Preprocessing.
- B. Primitive Volume Details.
- C. Quad Trees.

Preprocessing by the 3D Modelling System.

Given an Object, scene, or view definition, perhaps with a repeated sub-object deformed in different frames, the modelling system constructs an Object-tree, with each node describing a separate object instance, and the leaves being the primitive volumes. The nodes provide a place to store information about each object. For example, sequences of linear transformations between coordinate frames are combined. Another example; at the leaves of the Object-tree for a view, the coordinate patches and limbs of the surface are represented.

There are two more things that the modelling system does before it can be used (e.g. by the graphics program). First it finds the common intersection lines on the surfaces of intersecting objects. At present this information is given by hand. Then it works out which surface areas of the primitive volumes of intersecting objects actually lie on the surface of the composite object. It implements the formulas:

$$\partial(a \cup b) = ((\partial a \cap (\text{Neg}b)) \cup (\partial b \cap (\text{Neg}a)))$$

$$\partial a \cap b = ((\partial a \cap b) \cup (\partial b \cap a))$$

where *Neg* means negative volume, to decide which side of each intersection line is on the outer surface. The intersection lines are stored in the coordinate patch quad-trees with the outer surface on the left.

4.2.B. Primitive Volumes.

The basic building blocks are simple parametrized volumes. They have three parameters, (t, s, r) with each volume point $P = P(t, s, r)$ having unique parameters. $\partial P / \partial t$, $\partial P / \partial s$, $\partial P / \partial r$ form a right handed (not necess. orthogonal) set.

P is continuous in its parameters, and piecewise differentiable. The surfaces of discontinuity (called *jump-surfaces*) must be planar in parameter space. The purpose of this restriction is that it makes it easy to interpolate where a line segment in parameter space has crossed a jump-surface and to work out the closest jump-surface. At present these surfaces must be parallel to a parameter plane. To allow more general volume discontinuities and surface edges, arbitrary planar jump-surfaces could be implemented using a Kd-tree structure.

The bounds of the volume are defined by restricting the parameters t, s , and r to all lie in some volume of parameter space, typically the unit cube. So there is a mapping (position function P) from the unit cube of parameter space to real coordinate space.

Opposite faces and edges of the cube can be identified with each other, e.g. when one of the parameters is an angle, in cylindrical, or spherical coordinates.

The remaining faces of the cube map onto the surface of the volume. They have separate quad-tree representations where they store special points, regions, and lines, such as limbs or curves of parabolic points. Each face will form a coordinate patch, with one parameter fixed, and the other two parametrizing the surface. This is true for reasonable parametrizations.

So there are two kinds of edges on the volume surface,

1. where patches meet.
2. where jump-surfaces (derivative discontinuities) intersect the parameter cube.

4.2.C Quad Trees.

The condition for splitting squares up is that the opposite ends of each line segment stored lie in different areas. When testing for crossing of different segments, the longer one is split up further to the level of the shorter.

At the nodes, the segment end points are ordered around the borders of the square. This means that n segments in a square can be tested for intersection with each other in $O(n)$ time.

Also, regions of the tree can be marked by stepping around adjacent squares, moving away from the inside of the directed border segments.

Part Two.

When trying to unravel the structure of the projection mapping there are two questions.

1. What are the singularities and structure of the mapping

$P1$: surface \rightarrow image ?

This is the problem addressed by the first part of the paper. The *tooth-pick* representation summarises the qualitative structure of $P1$, while providing a framework for the quantitative details. Another way of thinking of $P1$ is,

$P1$: ray \rightarrow image, where one ray shoots out of the eye down the line of sight and intersects the surface in a number of places. The ray has two degrees of freedom (d.o.f.) Restricting it by one dimension, gets the generic limb singularities; by another, gets the generic point singularities of cusp (local) and T junction (not local).

$P1$: ray \rightarrow image, 2 d.o.f.

$P11$: limb-ray \rightarrow image, 1 d.o.f.

$P111$: cusp-ray \rightarrow image, 0 d.o.f.

$P112$: T-ray \rightarrow image, 0 d.o.f.

2. How do the singularities change for

(i) varying viewpoint, $P2$: Viewpoint $\rightarrow P1$,

(ii) varying surface shape, $P3$: Surface-Shape $\rightarrow P2$?

The singularities of $P2$ are roughly derived in the next section; here the three types of 2D signal singularities are summarized. Each has a generating curve and directions which sweep out the ruled surface of viewpoint singularities.

1. Line of parabolic points, with asymptotic direction at each point.
2. Line of asymptotic inflexions, with (2) asymptotic directions at each point.
3. Pair of T generating lines, where corresponding points, one from each line, have a common tangent plane. The special direction is along the line joining corresponding points.

The simplest singularities of $P3$ occur when the generating lines of $P2$, appear, merge or split.

5.1. Singularities for Changing Viewpoint.

As viewpoint changes the quantitative details of $P1$ change, but its structure is invariant over a range of viewpoints. $P1$ is stable for a generic viewpoint with 3 degrees of freedom. Restricting the viewpoint by one degree of freedom, there are three ways the structure of $P1$ can be altered.

1. (a) Limbs in the same limb set (i.e. surrounding a forward or backward facing region of surface), touch and merge, joining up two regions that face the same way.

(b) A new limb loop is born at a point.

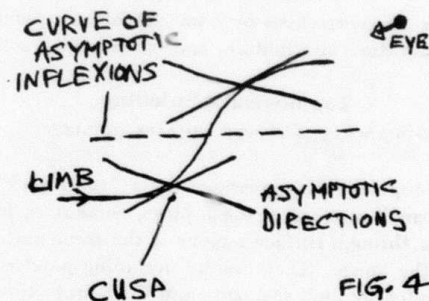
Both of these imply that the derivative of (surface-normal \cdot line-of-sight) is 0 in all directions at that point. The only way that this can be satisfied is to have one principal curvature zero, with its asymptotic direction coincident with the line of sight. So the singular viewing surface is ruled through the surrounding space, by sweeping a line tangent to the asymptotic direction along the closed loop, parabolic curves on the surface. This gives the mapping,

$P21$: viewpoint-on-swept-surface-1 $\rightarrow P1$.

The asymptotic direction is not necessarily tangent to the parabolic curve. Generically, one end points into the hyperbolic region, the other into the elliptic region. Looking along the asymptotic ray from elliptic to hyperbolic side, (b) is seen; the reverse direction gets (a).

2. Two cusps and a T junction appear in a fishtail shape, at a point on a limb in a hyperbolic region.

The condition for this is that the line of sight is tangent to one asymptotic direction at a point where the asymptotic line has an inflexion.



To derive this result, (see fig.4), imagine that the viewpoint is moving (with 3 d.o.f) so that the two cusps approach each other and merge. Since their limbs are tangent to the asymptotic direction, the cusps must approach by sliding in along that direction, which gives the asymptotic line an inflexion point where they meet.

So the singular viewing surface is ruled by sweeping a line in the asymptotic direction along the curve of asymptotic inflexions. This curve may intersect itself or intersect the curve of asymptotic inflexions of the other asymptotic direction, or be tangent to the curve of parabolic points. e.g. the curve around the middle of the hole in a torus. This gives,

$P22 : \text{viewpoint-on-swept-surface-2} \rightarrow P1.$

3. A pair of T junctions appear at a point.

Two contours become tangent in the image, and then form a pair of Ts. In order to see this singularity, the two surface points, whose contours are tangent, must have coincident surface tangent planes, with the line of sight aligned through both points.

$P23 : \text{viewpoint-on-swept-surface-3} \rightarrow P1.$

Pairs of T junction generating points form into two parallel closed loops. Each point on the upper loop has a corresponding point on the lower loop with a coincident tangent plane, and direction pointing to its mate. The singular surface is ruled by sweeping the direction around the loop.

There are several interesting 1 d.o.f viewpoint singularities and some weird 0 d.o.f. ones, where the viewpoint has to be at one special point in space. See [1] (pages 145 - 149) for a complete classification.

e.g. $P211 : \text{viewpoint-on-straight ray} \rightarrow P1$, where the ray is the asymptotic direction at a special parabolic point which has parabolic curve tangent to asymptotic direction. This is the borderline between 1(a) and (b).

5.2 Singularities for Changing Surface Shape.

Leaving aside for now the problem of how the structure of P2 is to be represented, what are the singularities of the mappings

$P3 : \text{Surface-Shape} \rightarrow P2,$

$Q3 : \text{Surface-Shape} \rightarrow P1 ?$

Some of the local singularities are:

- 1(a) A parabolic curve appears at a point. e.g. stretching and bending a sphere.
- (b) A parabolic line pinches off to form two parabolic lines. eg. turning a banana into a dumbbell. This can only happen when the two points that approach each other, where the split occurs, both have asymptotic direction tangent to the parabolic line. The two points annihilate each other at the split. (They are a parabolic line analog of cusps on limbs.)
2. Lines of asymptotic inflexion appear and split. Their cusp analogs are, points of self intersection, of intersection with other curves of asymptotic inflexion, and where they are tangent to a parabolic line.

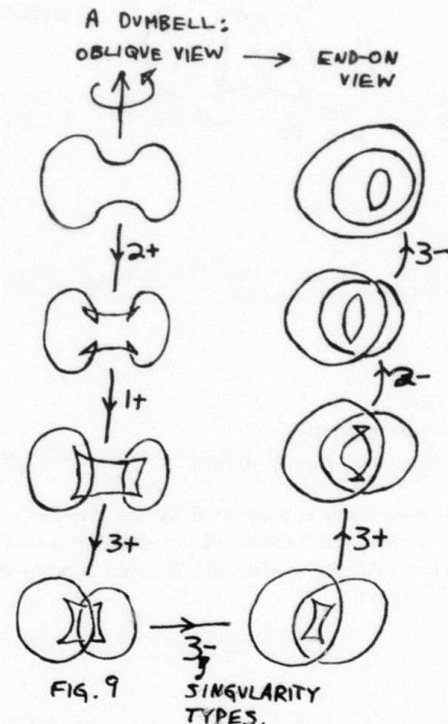
3. Similarly the T generators appear, merge and split.

6 Prediction Structures.

In [5], Koenderink and van Doorn describe a graph structure called the "Visual Potential" which predicts what an object looks like from its different characteristic views. Each node represents a volume in the viewing space, with a link between adjacent volumes. These volumes are sliced out of space by the ruled surfaces described above, and each link corresponds to a particular event on the object surface. E.g. limbs meet at some point on a segment of a parabolic curve.

A graph like this provides a general framework for 3D predictions.

To be useful for recognition or animation, it is not necessary to generate the whole graph; it is enough that at any viewpoint the structure of the graph locally can be generated. This is analogous to not generating the whole image, but only being able to work out what the image looks like near a single ray. At a node, something like the "tooth-pick" structure could be stored; along with the likely singularities (3 different types), so that links can be generated in response to a change in viewpoint. E.g., see fig. 9 for the sequence of views of a "dumbbell", moving from an oblique to an end-on view.



The singularities produced by changing the surface shape suggest a model hierarchy. One object might be described in terms of another by giving the sequence of singularities that occur when its surface is deformed to fit the other's. E.g., Forming a "dumbbell" (see fig. 10) by deforming a sphere. First the sphere is bent into a banana shape when a parabolic curve, bisected by a curve of asymptotic inflexions appears at a point on the sphere's surface. Then the two ends of the asymptotic inflexion curve, which are also on the parabolic curve, meet up around the girth of the banana. The parabolic curve splits into two, and the banana becomes a dumbbell.

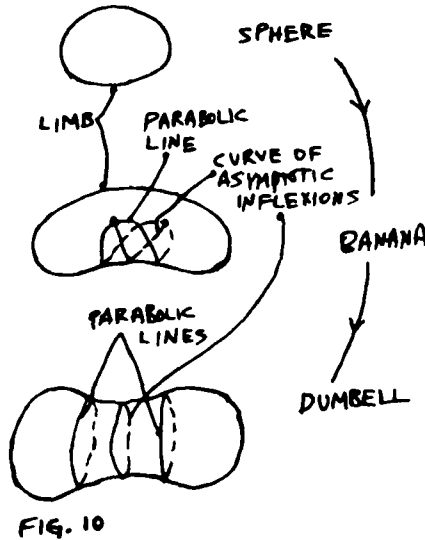


FIG. 10

References.

- [1] Arnol'd, V.I., "Singularities of Systems of Rays," *Russian Math Surveys*, **38:2** (1983), 87-176.
- [2] Binford, Thomas O., "Figure/Ground: Segmentation and Aggregation," *Proceedings: Rank Prize Fund, Conference in England*, 1982.
- [3] Brooks, Rodney a., and Thomas O.Binford, "Interpretive Vision and Restriction Graphs", *First National AAAI Conference*, 1980.
- [4] Koenderink, J.J., and A.J.van Doorn, "The Singularities of the Visual Mapping", *Biological Cybernetics*, **24** (1976) 51-59.
- [5] Koenderink, J.J., and A.J.van Doorn, "The Internal Representation of Solid Shape with Respect to Vision", *Biological Cybernetics*, **32** (1979) 211-216.
- [6] Hornung, Cristoph, "An Approach to a CalculationMinimized Hidden Line Algorithm," *Comput.& Graphics* **6:3** (1982) 121-126.
- [7] Scott, Richard S.F., "An Algorithm to Display Generalized Cylinders," *Proc. IU Workshop*, April 1983.

Acknowledgements.

Many thanks to Tom Binford.

This research is supported by the Air Force Office of Scientific Research F49620-82-C-0092, and previously by the Defense Advanced Research Projects Agency N00039-84-C-0211.

The Information-Centered Approach to Optimal Algorithms Applied to the 2-1/2 D Sketch

John R. Kender
David Lee

Department of Computer Science
Columbia University, New York, NY 10027

1. Abstract

In this paper, we introduce the *information-centered theory of optimal algorithms* as an approach to image understanding problems, and apply it to obtain a dense depth map from a sparse one (the "2-1/2 D Sketch"). There are three major results. First, we give a spline interpolation algorithm that is provably optimal in the worst case for surface reconstruction; since it is linear in the data, it is simple to compute using precomputed coefficients. Secondly, we show that adaptive information (that is, the intelligent and selective determination of which depth points to sample based on the values of previously sampled points) does not improve accuracy performance; a simple regular grid is provably optimal. Third, we discuss designs for implementations exploiting the above results which are very amenable to parallel processing, and allow for local, point-wise determination of surface character without the necessity for global optimization. We conclude with some remarks on our construction and execution of a preliminary form of one such implementation.

2. Introduction

The calculation of a full depth map of a scene from information present in an image is a central problem in image understanding. In general, what is desired in the full depth map is some "best" surface that fits the sparse and errorful depth data (the 2-1/2 D sketch) derived from shading, binocularity, motion, texture, and other "shape-from-x" surface cues. Mathematically, this can be cast as an interpolation problem subject to some error criterion. Much work has already been done [3, 4, 5]. This paper investigates the general problem from a different and relatively new viewpoint. We attempt to answer the following class of questions:

1. What algorithms are provably optimal with respect to the accuracy of the constructed full depth map?
2. What information on which to base the construction is provably optimal with respect to accuracy? (Here, which depth samples, taken in whatever place, in whatever order, however intelligently?)
3. What properties does the optimal algorithm have? Do the properties lead to feasible and, even, parallelizable computation?

We address the first question in Section 3, and show that spline algorithms are optimal with respect to the worst case error criterion. In Section 4, we first show that adaptive information, which is seemingly much more powerful than nonadaptive information (and certainly more computationally complex), does not improve accuracy performance. Thus one only has to seek for optimal information among the classes of nonadaptive information. In Section 5, we construct the spline algorithm. Spline algorithms are linear in their data, and hence favorable for parallel computations.

A note to the reader. In brief, our intention is to show how the problem of transiting from a sparse depth map to a full one can be cast in the framework of the theory of computational complexity and optimal algorithms. Once the problem is posed in the context and terminology of that field, the solution is a straightforward special case of several existing theorems. However, since the methods and terms of that subject are probably foreign to most vision researchers, we also take care in what follows to explicate, step-for-step, the reasoning behind the procedures, explaining the more abstract constructs in terms of the actual vision problem at hand. We therefore adopt the General Theory of Optimal Algorithms [6] to binocularity in what follows below, retaining much of the specialized notation, but with running glosses. In part, our intention also is to alert the vision community to the relevance of the research in this area. Additionally, we hope to exploit the theory further in later papers for other vision problems, such as optimal surface recovery from a monocular intensity array (optimal shape from shading).

3. What is the Optimal Algorithm?

The analysis proceeds in several steps. To begin, the problem must be restated as a problem of: classes of functions (here, of three-dimensional surfaces), available information, and classes of algorithms. The following aspects must be quantified, and we discuss them in turn:

1. The space of surfaces.
2. The information available and the dependencies by which it is obtained.
3. The class of algorithms.
4. The measure of error and the meaning of "optimal".
5. The specification of splines and spline algorithms.
6. The optimality of spline algorithms.

3.1. Choosing the Space of Surfaces and Their Norms

We take as our space of (possibly infinite) real-world surfaces the following class:

Let F be the set of all real-valued functions f defined on R^2 , such that f and its first and second order partial derivatives all belong to $L_2(R^2)$. That is, the class of real-world surfaces is smooth at least up to local curvature: their curvatures are square-integrable. In particular, this class rules out any surfaces that are merely piece-wise continuous or differentiable. These two latter exceptions, unfortunately, rule out true occlusions (where depth is discontinuous) and true corners (where the derivative is). Thus, the world to be seen appears as if it were shrink-wrapped: corners are rounded and discontinuities papered over. Inasmuch as the surfaces of objects tend to be locally smooth, however, this appears to be a reasonable assumption.

We attempt to "see" a subimage of these real-world surfaces. They are supported in a finite region D of the xy plane; visually speaking, it is the xy plane that forms the background, and D that forms a finite subimage. For simplicity, assume that D is a square. Then the class of surfaces F_2 that we want to recover is the constriction of the surfaces in F_1 to D :

$$F_2 = \{ f \mid D : f \in F_1 \} \quad (1)$$

3.2. Quantifying the Idea of Information

To recover the surface (a member of F_2), we start with samples of depth data (in region D) of some type; these samples we would normally call *information*. More precisely, *information* is defined in the general theory as a *function* of the following form.

$$N : F_1 \rightarrow R^k$$

That is, each type of information function is a mapping from the class of given surfaces to k -vectors of image primitive values. Each f (each real-world surface) in F_1 is a member of the domain of N ; $N(f)$ is the vector of samples. In terms of the vision problems, a given $N(f)$ tells in what way a smooth surface, f , has been captured into a k -vector of extracted image primitives. (Thus, the theory uses "information" to make more precise the concept of "intrinsic image"; information can be velocities, surface orientations, etc.) In the simplest case, and the case in point here, N would be the rule for extracting depth values themselves:

$$N(f) = [f(x_1, y_1), f(x_2, y_2), \dots, f(x_k, y_k)], \quad (2)$$

$$(x_i, y_i) \in D, i = 1, \dots, k.$$

For ease of analysis, we require that $k > 3$, and further, that the k depth values not be coplanar. This rules out annoying special cases which, although they are easily handled, would otherwise have to be explicitly addressed in much of what follows.

We can generalize the sampling of these data values a bit, by writing:

$$L_i(f) = f(x_i, y_i), i=1, \dots, k. \quad (3)$$

One can easily check that each $L_i(\cdot)$ is a linear functional on F_1 . Our information is now in the form about which most is known in the general theory:

$$N(f) = [L_1(f), L_2(f), \dots, L_k(f)], i=1, \dots, k. \quad (4)$$

In requiring the components of $N(f)$ to be linear functionals, the vector $N(f)$ is implicitly restricted to be a collection of samples taken "in parallel" from f at points that can be predetermined independently. That is, the i -th component of $N(f)$ depends only on f , rather than on some dynamically changing sampling method based on the previous ($i-1$) components of $N(f)$. In short, this information is *nonadaptive*. It is in contrast to the information used in many optimum-seeking algorithms (such as root-finding), which selectively sample promising areas increasingly more densely, based on their nearness to an optimum.

Superficially, an implicit restriction of $N(f)$ to nonadaptivity seems to be a restriction to a less powerful set of information-gathering strategies. It will turn out, however, that it has absolutely no effect. Even if we extend the definition of information to allow the use of *any* adaptive sampling of linear functionals, no matter how intelligent, the intrinsic error is no less than that obtained by using nonadaptive information only. (We will address this issue in Section 4.) Thus, what is important in terms of the general theory is simply that the information be *linear*, such as depth values of surfaces are.

3.3. Defining the Class of Algorithms

From the information $N(f)$, which is a k -vector of samples (here, of depth values), we now choose an *algorithm* ϕ to recover f in F_2 (here, that part of the real-world surface we attempt to "see"). The algorithm ϕ is defined in a very general way as a member of a class of mappings as follows:

$$\phi : R^k \rightarrow F_2, \phi \in \Phi. \quad (5)$$

Thus, in the general theory an algorithm maps vectors of samples (of a function f in F_1) into a solution function (in F_2). Note that in this general definition, F_1 and F_2 are usually different classes. For example, in classical quadrature algorithms for numerical integration, F_1 is the class of functions to be integrated, $N(f)$ are sample points, ϕ is the quadrature formula, and F_2 is simply R^1 , the set of reals. Here, F_1 are real-world surfaces and F_2 are their (restricted) reconstructions over D .

3.4. Defining Algorithm Error, Optimal Algorithm

In order to compare solutions and to measure the approximation error of ϕ , F_2 can be equipped with a norm, $\|\cdot\|_2$. By applying this norm, $\|f|D - \phi(N(f))\|_2$ now quantifies the difference between $f|D$, which is that portion of the real-world surface to be recovered over D , and $\phi(N(f))$, which is the surface we construct by using information N and algorithm ϕ .

There are many choices for $\|\cdot\|_2$; even a weak one will do. For our problem of depth interpolation, we can use the L^2 -norm, defined as:

$$\|f\|_2 = \left\{ \int_D f^2 dx dy \right\}^{1/2} \quad (6)$$

Which norm is the most natural or accurate measure of algorithm error? It turns out that the determination of the optimality of the algorithm is independent of the choice.

We are nearly ready to define the error of a given algorithm. Based on this definition, we will seek an algorithm that reconstructs the surface with minimum error. We would like to define the error of a particular algorithm in the worst case to be something like the following:

$$e(N, \phi, f) = \sup \|f|D - \phi(N(f))\|_2 \quad (7)$$

where the supremum is taken over all f in F_1 that satisfy the same information. That is, the supremum should be taken over the set $V(N, f)$, where

$$V(N, f) = \{f' \in F_1 \mid N(f') = N(f)\}. \quad (8)$$

This definition measures the distance between the actual computed solution, $\phi(N(f))$, and all functions f' in F_1 that could possibly have been the source of the observed information. Since the f' in $V(N, f)$ are completely indistinguishable (we know nothing besides the information $N(f)$), we cannot tell which of them could have been the original function. Thus, we take the supremum to handle the worst case.

The problem with such a definition is that the class of functions in F_1 is usually too large. Given $N(f)$, there are infinitely many interpolating f' in $V(N, f)$. Unless the 2-norm is, in a sense, very weak, for many of these functions, the supremum and the worst case error will almost certainly be very large. However, in terms of the physics of the image understanding problem, many of these surfaces would also be physically impossible as well: some would have to pass through the camera itself; others would be impossible to fabricate under any known natural or artificial manufacturing process.

What we usually prefer instead is a solution function that comes as close as possible to "reasonable" members of $V(N, f)$, rather than to all of them. Intuitively, a function is considered "reasonable" if it satisfies some desirable side conditions. Mathematically, such a function is often characterized by expressing the desired properties in terms of another norm (or semi-norm), and defining "reasonable" to mean that these values are within certain bounds. We will denote the reasonableness norm by the 4-norm; just as there are many choices for the 2-norm, the actual definition of the 4-norm is determined by the problem.

One example of a desirable property for functions in $V(N, f)$ is smoothness; this can be quantified by applying to elements of F the quadratic variation semi-norm. This semi-norm is defined to be:

$$\|f\|_4 = \left\{ \int_{R^2} [(f_{xx})^2 + 2(f_{xy})^2 + (f_{yy})^2] dx dy \right\}^{1/2} \quad (9)$$

Given that second derivatives are closely related to surface curvature, this semi-norm has an appealing intuitive physical analogy: it measures the bending energy in an ideally thin and elastic plate which has been forced into the shape of f . We will return to this example when we discuss our efforts on implementation.

We can use this semi-norm (or any other "reasonable" one) to better define algorithm error. There are many ways to do so; one way would be, for each interpolating f , to divide the approximation error given by the 2-norm in Equation (7) by some function of the reasonableness of f . This gives a type of relative error for each f ; the more badly behaved a surface, the higher its curvature and the more vigorously its actual algorithmic error would be restrained. Note that even particularly wild functions are still required to interpolate the information, so that it is likely that their relative error is rather small; the calculation would be dominated by the extreme normalizing value.

We redefine algorithm error, then, in the general case as follows:

$$e(N, \phi, f) = \sup \|f|D - \phi(N(f))\|_2 \rho(\|f\|_4) \quad (10)$$

where the supremum is again taken over all f satisfying the information N (that is, f is in $V(N, f)$). The reasonableness function ρ simply maps the positive value of the 4-norm into a new positive value: if $\rho(x) = 1/x$, this is simple relative error; if $\rho(x) = 1$, we have recovered absolute error again; many others are possible. Importantly, one of the results of the general theory is that optimality is independent of the exact nature of ρ .

Now we can define what an optimal (in the worst case) algorithm is. The algorithm ϕ^* in Φ , the class of algorithms, is strongly optimal if and only if for each f in F :

$$e(N, \phi^*, f) = \inf e(N, \phi, f) \quad (11)$$

where the infimum is taken over all algorithms ϕ in Φ . That is, an algorithm ϕ^* is optimal if and only if the algorithm error from using ϕ^* (for any f in F) is no more than the algorithm error from using any other ϕ in Φ . Note that this does not imply that the optimal algorithm must necessarily have finite algorithm error. However, if the optimal algorithm does have infinite algorithm error (that is, if at least one f has infinite approximation error), then every other algorithm has infinite algorithm error, too.

3.5. Spline Functions

We next define a particular interpolating function which is based on our reasonableness norm, $\|f\|_4$. Although this function is derived from what appears to be a problem-

dependent side condition, it will be the primary function that leads to the optimal solution of the interpolation as a whole.

Recall that $V(N, f)$ is the class of all functions in F_1 that share the same information with f under the information extraction function N . In purely visual terms, it is the set of all surfaces defined on the infinite background that interpolate the depth data. Let $\sigma_{N(f)}$ be that member of $V(N, f)$ with minimum 4-norm. That is,

$$\|\sigma_{N(f)}\|_4 = \inf \|f\|_4 \quad (12)$$

where the infimum is taken over all f in $V(N, f)$. We call $\sigma_{N(f)}$ the spline function interpolating the data $N(f)$. It is not hard to show that such a spline function exists and is unique, if there are at least four non-coplanar data points, which is true for almost all cases. For more details, see [7], page 71.

Thus, this unique spline function $\sigma_{N(f)}$ interpolates the information, and, because it minimizes the 4-norm, of all such interpolants it is the most "reasonable". This need not imply that it also minimizes worst case algorithm error; algorithm error is based on a different norm, in a more constricted space. However, it is surprising that under very general conditions, $\sigma_{N(f)}$ directly provides the optimal algorithm.

3.6. Spline Algorithms and their Optimality

Our last step is to define a class of algorithms based on the spline functions; this class will contain the optimally interpolating algorithm we seek.

The algorithm ϕ^s that takes information $N(f)$, chooses the interpolating spline function $\sigma_{N(f)}$, and then constricts it to D , we call a spline algorithm. More precisely, the spline algorithm is defined as:

$$\phi^s(N(f)) = \sigma_{N(f)}|D, \quad (13)$$

where $\sigma_{N(f)}$ is the spline function. In our example with quadratic variation as semi-norm, the spline is unique, so the spline algorithm is well defined.

The import of ϕ^s is given by the following theorem.

Theorem. The spline algorithm is strongly optimal (in the worst case). That is,

$$e(N, \phi^s, f) = \inf e(N, \phi, f), \text{ for all } f \in F_1. \quad (14)$$

where the infimum is taken over all ϕ in Φ . Further, the optimality of the spline algorithm is independent of the choice of norm $\|\cdot\|_2$.

The proof is based on the following observations. First, it can be shown that the definition of the algorithm error as defined in Equation (10) is equivalent to the following definition, which assigns the 4-norm a more tractable role:

$$e(N, \phi, f) = \sup \|f|D - \phi(N(f))\|_2, \quad (15)$$

where the supremum is now taken over the smaller class of information-satisfying functions that have only a strictly limited reasonableness: f must be in $V_4(N, f)$, where

$$V_4(N, f) = \{f \in V(N, f), \|f\|_4 \leq c\}. \quad (16)$$

The proof is given in [7], Appendix E; the finite value c is arbitrary. Essentially this observation allows one to define directly a class of "reasonable" functions, and to measure approximation error on an absolute basis using the 2-norm alone.

Secondly, the spline function $\sigma_{N(f)}|D$ can be shown to be the exact center of the set $V_4(N(f)|D)$, and thus it must minimize the worst case approximation error. The centrality is proven by showing that every $f|D$ in $V_4(N(f)|D)$ can be expressed as the sum $\sigma_{N(f)}|D + h$, where the properties of h are sufficient to show that the difference $\sigma_{N(f)}|D - h$ is also in $V_4(N(f)|D)$.

For the detailed proof of the entire theorem, see [6] Theorem 5.1, page 76.

4. Adaptive Information Does Not Help

In Subsection 3.2, we defined information as samples of depth data:

$$N(f) = [L_1(f), L_2(f), \dots, L_k(f)], \quad i=1, \dots, k, \quad (17)$$

where $L_i(f) = f(x_i, y_i)$, and $(x_i, y_i) \in D$. We call this nonadaptive information, since the i -th component of $N(f)$, $L_i(f)$, depends only on f . Adaptive information, on the other hand, attempts to exploit whatever was learned while obtaining the $(i-1)$ components of $N(f)$. More precisely, adaptive information N^* is

$$N^*(f) = z = [z_1, \dots, z_k] \quad (18)$$

where $z_i = L_i(f, z_1, \dots, z_{i-1})$, $i=2, \dots, k$. In the case of depth values,

$$z_i = f(x_i, y_i) \quad (19)$$

where $x_i = x_i(z_1, \dots, z_{i-1})$ and $y_i = y_i(z_1, \dots, z_{i-1})$, with $(x_i, y_i) \in D$.

The structure of adaptive information is much richer than nonadaptive information, and one might hope that, by virtue of adaption, some intelligence might determine the location for (x_i, y_i) on the basis of the results of the $(i-1)$ prior samplings. Nevertheless, theory shows that, against intuition, adaptive information cannot aid surface approximation. For detailed discussions and proof, see [7], pages 57-62. The formal proof is based on the radius of information. Intuitively, the radius estimates the intrinsic error of the problem. For several classes of problems (including interpolation), the radius cannot be reduced by adaptive strategies, in large part because there exist fixed but universal strategies. This is perhaps the strongest result of all: one cannot do better in collecting data than a snapshot does. Not only can the data be collected in parallel, it should.

At this point, we have shown that the spline algorithm is the optimal interpolation algorithm, and that nonadaptive information suffices. However, we have not attempted to optimize where to obtain the information itself. It is apparent that all sampling strategies are not equal. If we are free to select the location of information points, what points are optimal?

Restating this problem, suppose we are allowed to choose a k -vector of information samples: $N(f) = [f(x_1, y_1), \dots, f(x_k, y_k)]$. Suppose, too, that no matter what information we select, we always use the optimal spline algorithm. Since the algorithm is tailored to the information, any error that would remain is intrinsically irreducible. We then can define optimal information, denoted by N^* , to be that information with the minimum intrinsic error.

For our depth map problem, the optimal choices for (x_i, y_i) can be shown to lie on a regular grid. More precisely, let the subimage D be the open rectangle: $(0, (n+1)h) \times (0, (n+1)h)$. Then the following information N^* is optimal (up to a constant factor) for surface recovery:

$$N^*(f) = [f(h, h), f(h, 2h), \dots, f(h, nh), \dots, f(2h, h), f(2h, 2h), \dots, f(2h, nh)] \quad (20)$$

$$\dots, f(nh, h), f(nh, 2h), \dots, f(nh, nh)]$$

These are simply interior mesh points. Notice that the optimality of this information (and, of course the resulting error of the optimal algorithm using this optimal information) depends on the norm $\| \cdot \|_2$; in general, the intrinsic error is monotonically decreasing in h . For a full proof of the optimality of this particular N^* mesh, see [1].

5. Implementation of the Optimal Algorithm

In the previous sections we have shown the existence and uniqueness of the spline interpolating given depth data, and we have shown its optimality for surface recovery problems. In this section, we show how the spline functions can be constructed, with the side condition of minimizing quadratic variation. Note that in what follows, we do not necessarily require optimal information; the depth samples can appear anywhere within the subimage D : in a mesh, aligned on contours, clustered, or even at random.

With some restrictions on F_{11} , it can be shown that the appropriate reproducing kernel is $K(x, y; u, v) = \{(x-u)^2 + (y-v)^2\}^{3/2}$, the Euclidean distance cubed (see [2]). Therefore, the spline interpolating depth data $z = [z_1, \dots, z_k]$ is developed in the usual way:

$$\sigma_z = \sum_{i=1}^k \alpha_i \{(x-x_i)^2 + (y-y_i)^2\}^{3/2} + \beta_1 x + \beta_2 y + \beta_3 \quad (21)$$

where $\{\alpha_i\}$ and $\{\beta_i\}$ can be determined from the linear system of equations:

$$\begin{aligned} \sum_{i=1}^k \alpha_i \{(x_j - x_i)^2 + (y_j - y_i)^2\}^{3/2} + \beta_1 x_j + \beta_2 y_j + \beta_3 &= z_j, \\ \sum_{i=1}^k \alpha_i x_i &= 0, \\ \sum_{i=1}^k \alpha_i y_i &= 0, \\ \sum_{i=1}^k \alpha_i &= 0. \end{aligned} \quad (22)$$

From equations (21) and (22), it should be apparent that the splines are linear in the data. That is, if σ_1 interpolates information $z^{(1)}$, and σ_2 interpolates information $z^{(2)}$, then $c_1 \sigma_1 + c_2 \sigma_2$ interpolates information $c_1 z^{(1)} + c_2 z^{(2)}$. In terms of image understanding, this means that if two surfaces are superimposed so that their depths samples accumulate, then the superimposition of the two full depth maps derived independently create a valid full depth map for the ensemble.

Since the spline algorithm is linear in depth data, it can be easily rewritten as the weighted sum of basis splines, as follows. Suppose that the information is merely i -th unit vector for R^k , that is, $N(f) = e_i = [0, \dots, 0, 1, 0, \dots, 0]$, where the unit is in the i -th coordinate position. This simpler information constraint is satisfied by a unique basis spline function σ_i , with the property that $\sigma_i(e_i) = \delta_{ii}$ (the Kronecker delta). In terms of the depth interpolation problem, σ_i generates a surface that has a value of 1 at sample point (x_i, y_i) , is identically zero at all other sample points, and is smoothly rippled in all the space between, in order to minimize its bending energy. The spline

interpolating depth data $z = [z_1, \dots, z_k]$ then is simply the weighted sum of these individual basis splines, with z -values as the weights:

$$\sigma_s = \sum_{i=1}^k z_i \sigma_i \quad (23)$$

Therefore the problem of computing σ_s may be reduced to that of solving the k independent subproblems of computing each σ_i . This has several important consequences for implementation.

1. Since the heart of the method is the solution of a system of linear equations, much is known about its convergence, stability, and running time (about $O(k^3)$).
2. If the location of the information is known beforehand, the basis splines can be precomputed.
3. With precomputed coefficients, the desired interpolation points can be computed in parallel with a simple SIMD algorithm.
4. With precomputed coefficients, any individual value of the solution surface can be determined locally, without the need for global optimization over the full surface: if just one is needed, just one is computed.
5. With precomputed coefficients, the surface can be incrementally updated. Any sample value that changes over time has only a linear effect on existing interpolated data. The increments at each point can be computed in parallel with a trivial SIMD algorithm.
6. If one uses optimal information, the system of coefficients becomes highly regular; the matrix has only a very small number of distinct entries (for large k , approximately $k/2$ for a k -by- k system). Symmetries suggest that efficient solution is possible, even without precomputation.
7. If one uses optimal information, the system shows an eight-fold symmetry, which could probably be exploited both in precomputation and execution.

6. Preliminary Experimentation

These results suggest that it would not be hard to construct a special-purpose machine for surface interpolation that would be very quick and accurate. Using active imaging, it could obtain depth samples on a square grid of k total points, by ranging or by triangulation. The position of these sample points would remain fixed, so all coefficients could be precomputed. Run-time computation would entail only the distribution of input data and the calculation and collection of output data, using weighted sums of precomputed coefficients. Thus, the interpolated values at any point (x,y) are given by:

$$\sigma_s(x,y) = \sum_{i=1}^k z_i \sigma_i(x,y), \quad (24)$$

where $z = [z_1, \dots, z_k] = [f(x_1, y_1), \dots, f(x_k, y_k)]$ and each σ_i has been previously computed. Such a SIMD algorithm would require about k multiplications, plus k units of local storage per process. If special purpose hardware were available, the data could be circulated in a type of toroidal systolic array. All output would be complete in roughly k cycles. Precomputation could be achieved in k parallel streams as well.

We have simulated some of this behavior on a standard uniprocessor. We briefly list the following experimental preliminary results, obtained by Terry Boulton. Many of them suggest there may be algorithmic or computational efficiencies to be exploited.

1. Depending on how one enumerates the k points, under optimal information the Gram matrix (the matrix of the cubic distances which one solves to obtain the coefficient matrix) appears to have a recursive block Toeplitz structure.
2. Experimentally, this matrix appears to be well-conditioned with respect to computing its inverse, as fairly large systems ($k=100 \times 100$) show little loss of precision: coefficients known to be symmetric or equal retain their symmetry and equality to about the limits of ordinary round-off error.
3. The inverse of the Gram matrix is highly sparse; when $k=100 \times 100$, there are only 621 distinct entries. Further, the ratio of distinct entries to total entries appears to decrease as k increases.
4. Although the basis splines do not have compact support, they appear to fall off rather rapidly.

7. Related Problems in Computer Vision

The theoretic results reported above--the optimality and linearity of spline algorithms, the sufficiency of nonadaptive information, and others--apply to other vision problems that can be cast in the same fairly loose framework. The theory requires that F_1 be an arbitrary linear space, and that the information be a vector of linear functionals on F_1 . In terms of vision, the linearity of F_1 is rarely a problem, since object surfaces superimpose well; however, only some classes of image features can be considered to be linear information. The trouble is that linear information must also superimpose: the features derived from "sum" of two surfaces must be the sum of the features independently derived. This is often contrary to the laws of geometry, physics, and optics; shading clearly does not sum well. Nevertheless, there are several important types of linear vision information, among which are:

1. The depth values themselves, at any place in the image: $L_1 = f(x_i, y_i)$. This is the problem just analyzed. Note that there need not be any restriction on the location of (x_i, y_i) ; they can even be chosen randomly.
2. The depth values derivable from a contour: $L_1 = f(x_i, y_i)$, where the values are restricted to lie on a particular curve. This is the usual result of the first stage of edge detection methods (zero-

crossing contours, etc.), or work on silhouettes.

3. A given directional derivative: $L_i =$ the directional derivative of $f(x_i, y_i)$ with respect to l_i , where l_i is a direction vector in R^2 . This would be a one-dimensional variant of the shape-from-shading problem, where the available information samples are the uniquely determined surface slopes in a particular direction. (A richer formulation would acquire the two-dimensional surface orientation by means of the gradient vector. A richer and more practical formulation would extend the definition of information to capture constraints on surface normals, such as those generated by shading.)

The linearity of information is important: it appears to be a key determinate of many of the existing results of the general theory. Inasmuch as many image observables are non-linear functionals of the surface, these important cases of non-linear problems remain to be thoroughly treated. Work continues: recently, it has been proved that non-linear continuous information is not more powerful than linear continuous information [8]. Additionally, although most results deal with worst case models, the average and asymptotic cases are also under investigation: see [7], page 85.

8. Future Work

We see several areas of great interest. We plan to investigate the effects of missing or errorful information. The general theory is being pursued along those lines as well, so some results may straightforwardly fall out.

More practically, perhaps our most pressing interest is in finding an efficient algorithm for evaluating the basis spline coefficients. As an aid, we are pursuing the idea of displaying the matrix as an *image itself* to get a better understanding of its structure. Again, since an exact solution may be difficult, we are also exploring various approximate techniques, particularly with regard to replacing the basis splines with ones that are finitely supported, or with ones that are only asymptotically correctly shaped for their position.

9. Summary

We believe that the information-centered approach to algorithms can be applied to many vision problems with powerful results. In this paper, we introduced the method and have shown how results pertinent to depth map interpolation are corollaries of the general theory. The major results are that spline interpolations are provably optimal in the worst case, that the resultant linear algorithms are exceedingly simple and parallelizable given some precomputation, and that adaption does not help. Our hope is that the application of this approach to other vision problems will provide similar insight and computational power.

10. Acknowledgments

We thank J. F. Traub and G. W. Wasilkowski for their friendly and valuable comments.

References

1. Babenko K.I. (editor). *Theoretical Foundations and Construction of Computational Algorithms for the Problems of Mathematical Physics*. Moscow, 1979 (in Russian).
2. Duchon, J. "Interpolation de Fonctions de Deux Variables Suivant le Principe de la Flexion des Plaques Minces." *Revue Francaise d'Automatique, Informatique et Recherche Operationnelle* (Dec. 1976), 5-12.
3. Grimson, W. E. L.. *From Images to Surfaces*. MIT Press, 1981.
4. Ikeuchi, K. and Horn, B. K. P. "Numerical Shape from Shading and Occluding Boundaries." *Artificial Intelligence* 17 (1981), 141-184.
5. Terzopoulos, D. *Multiresolution Computation of Visible-Surface Representations*. Ph.D. Th., MIT Artificial Intelligence Lab., 1984.
6. Traub, J. F. and Wozniakowski, H.. *A General Theory of Optimal Algorithms*. Academic Press, 1980.
7. Traub, J.F., Wasilkowski, G.W., and Wozniakowski, H.. *Information, Uncertainty, Complexity*. Addison-Wesley, Reading, MA, 1983.
8. Wasilkowski, G.W. How Powerful is Continuous Nonlinear Information? Department of Computer Science Rep., Columbia University, 1984.

DETERMINING 3-D MOTION AND STRUCTURE FROM OPTICAL FLOW GENERATED BY SEVERAL MOVING OBJECTS

Gilad Adiv

Computer and Information Science Department
University of Massachusetts
Amherst, MA 01003

ABSTRACT

A new approach for the interpretation of optical flow fields is presented. The flow field, which can be produced by a sensor moving through an environment with several, independently moving, rigid objects, is allowed to be sparse, noisy and partially incorrect. The approach is based on two main stages. In the first stage the flow field is segmented into connected sets of flow vectors, where each set is consistent with a rigid motion of a roughly planar surface. In the second stage sets of segments are hypothesized to be induced by the same rigidly moving object. Each of these hypotheses is tested by searching for 3-D motion parameters which are compatible with all the segments in the corresponding set. Once the motion parameters are recovered, the relative environmental depth can be estimated as well. Experiments based on real and simulated data are presented.

1. INTRODUCTION

Dynamic visual information can be produced by a sensor moving through the environment and/or by independently moving objects in the visual field. The interpretation of such information consists of dynamic segmentation, recovering the motion parameters of the sensor and each moving object, and structure determination. The results of this interpretation can be used to control behaviour, as in robotics or navigation. They can also be integrated, as an additional knowledge source, into an image understanding system, such as the VISIONS system [HAN78].

The most common approach for the analysis of visual motion is based on two phases: computation of an optical flow field and interpretation of this field. In the present discussion, the term 'optical flow field' refers to both a 'velocity field', composed of vectors describing the instantaneous velocity of image elements, and a 'displacement field', composed of vectors representing the displacement of image elements from one frame to the next. In the latter case we will assume small values of motion parameters.

The second phase, i.e., the interpretation of the optical flow field, is the main concern of this paper. A new scheme is proposed, which allows motion of the camera as well as

rigid objects in the scene. Furthermore, the flow field is allowed to be sparse, noisy and partially incorrect. The information in only one flow field, as opposed to a time sequence of such fields, is utilized.

Our approach is based on two main stages. In the first stage the flow field is segmented into connected sets of flow vectors, where each set is consistent with a rigid motion of a roughly planar surface. In the second stage sets of segments are hypothesized to be induced by the same rigidly moving object. Each of these hypotheses is tested by searching for 3-D motion parameters which are compatible with all the segments in the corresponding set. Once the motion parameters are recovered, the relative environmental depth can be estimated as well.

In the next section, techniques existing in the literature for visual motion interpretation are examined. The mathematical formulation of the model and the task is presented in section 3. In subsequent sections, algorithms for flow field segmentation, estimation of motion parameters, and structure determination are developed. Preliminary experiments based on real and simulated data are described in section 6.

2. LITERATURE REVIEW

In this section we review methods existing in the literature for interpreting optical flow fields. We concentrate on techniques which assume rigid motion and basically rely on the information contained in one flow field. Two main issues are emphasized:

a) **Scene Complexity.** Some researchers assume that the scene contains only one object, or, equivalently, that the sensor is moving but the environment is stationary (e.g., [BRU81], [LAW82], [TSA84]). Others allow the scene to contain several independently moving objects (e.g., [ULL79], [NEU80]).

b) **Robustness.** Optical flow fields produced from real images by existing techniques are noisy and partially incorrect (see the discussion in [ULL81]). Many of the algorithms described in the literature for interpretation of flow fields fail under such conditions. Other algorithms are less sensitive and work reasonably well on real world images.

In the first class of techniques, discussed in this re-

view, only one rigid object (or camera motion) is assumed. A few researchers [ROA80, PRA80, NAG81a,b, FAN83a,b] present sets of nonlinear equations with motion parameters as unknowns. Methods for solving such equations are usually iterative and require initial guesses of the unknowns. Sensitivity to noise is indicated by experiments reported in [ROA80, PRA80, FAN83a,b].

Longuet-Higgins [LON81] and Tsai and Huang [TSA84] develop techniques based on solving a set of linear equations. Furthermore, conditions for the uniqueness of the solutions are formulated. However, difficulties in the presence of noise are still reported [TSA84].

Bruss and Horn [BRU81] employ a least squares approach which minimizes some measure of the discrepancy between the measured flow and that predicted from the computed motion parameters. In the case of general rigid motion this approach leads to a system of nonlinear equations from which the motion parameters can be computed numerically. This method is computationally more complicated than the methods offered in [LON81] and [TSA84], but seems to be more robust in the presence of noise.

Assuming a purely translational motion, all the flow vectors are oriented towards or from a single point in the image plane. Determining this point, called the focus of expansion (FOE), yields the direction of the translation. A few techniques, reviewed below, are based on this observation.

Early results based on real images are reported in [WIL81]. However, only sensor motion restricted to translation is allowed and the environment is assumed to contain only planar surfaces at one of two given orientations. Thus, the algorithm can be based on a search for the FOE and the distances to the surfaces in the scene. Lawton [LAW82] describes a robust algorithm which has been applied to real world images from several different task domains. This algorithm requires no restrictions on the shape of the environment, but is still restricted to translation. It is based on a global sampling of an error measure corresponding to the potential positions of the FOE, followed by a local search to determine the exact location of the minimum value. Results for other restricted cases of motion are presented in [LAW84].

Praschny [PRA81] describes a method which relies on decomposition of the velocity field into rotational and translational components. For a hypothesized rotational component, the FOE of the corresponding translational field and a related error measure are computed. Thus, an error function of the 3 rotation parameters is obtained and the solution can be determined by minimizing this function. Jerian and Jain [JER83] report on difficulties with applying a similar approach to noisy data.

Rieger and Lawton [RIE83] develop a relatively robust and simple procedure for computing the motion parameters, based on the fact that the differences between optic flow vectors near occlusion boundaries are oriented towards the

FOE of the translational field. However, the environment is assumed to contain occlusion boundaries which endow the flow field with strong discontinuities.

A number of methods, presented in the literature, allow (at least in principle) unconstrained sensor motion and independently moving objects in the environment. Ullman, in his somewhat pioneer work [ULL79], examines small sets of adjacent vectors. If there exists a unique rigid interpretation consistent with all the vectors in a given set, then this interpretation is assumed to be correct and the vectors in the set are grouped together. This approach seems to be very sensitive to noise because of its local nature.

Longuet-Higgins and Prasadny [LON80] and Waxman and Ullman [WAX83] introduce equations for computing the motion parameters and the local structure at a given point in the environment from the flow field and its first and second spatial derivatives at the corresponding point in the image. If the scene consists of several objects in relative motion, then a separate computation can be carried out on each one. However, local estimates of the second derivatives of the optic flow seem to be inaccurate in the presence of noise, and no algorithm has been presented for reliably computing such derivatives.

More global approaches are proposed in [NEU80] and [BAL81b]. Neumann [NEU80] proposes an elegant hypothesize-and-test scheme: for any rotation hypothesis, the translation component may be decomposed such that motion compatibility of many flow vectors can be easily tested. This technique heavily relies on the assumption of orthographic projection.

Ballard and Kimball [BAL81b] apply the generalized Hough technique to the optical flow field and thus extract the motion parameters. This is a global approach which is relatively insensitive to noise. In principle, it can also be used in scenes containing independently moving objects. However, the depth information is assumed to be known, thus making the task much easier.

This review demonstrates typical constraints and weaknesses of algorithms reported in the literature. No algorithm for interpretation of optical flow fields in scenes containing several, independently moving, rigid objects, has been shown to work with noisy, real world data, unless severe constraints are assumed or additional information is utilized.

3. THE MODEL AND THE TASK — A MATHEMATICAL FORMULATION

3.1 Basic Model and Equations

In this section we present a notation for describing the motion of a camera through an environment containing independently moving objects. We also review the equations describing the relation between the 3-D motion model and the corresponding optical flow, assuming a perspective projection. The equations are developed both for velocity fields

and displacement fields.

Let (X, Y, Z) represent a cartesian coordinate system which is fixed with respect to the camera (see figure 3.1) and let (x, y) represent a corresponding coordinate system of a planar image. The focal length, from the nodal point O to the image, is assumed to be known. It can be normalised to 1, without loss of generality. Thus, the perspective projection (x, y) on the image of a point (X, Y, Z) in the environment is:

$$x = X/Z, \quad y = Y/Z. \quad (3.1a,b)$$

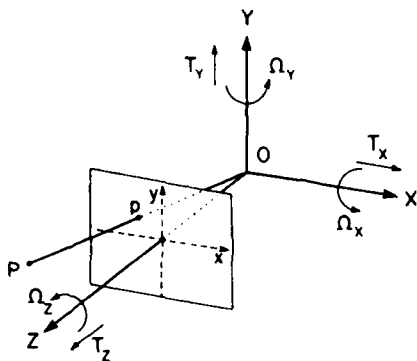


Figure 3.1(redrawn from [LON80]): A coordinate system (X, Y, Z) attached to the camera, and the corresponding image coordinates (x, y) . The image position p is the perspective projection of the point P in the environment. $\mathbf{T} = (T_X, T_Y, T_Z)$ and $\mathbf{\Omega} = (\Omega_X, \Omega_Y, \Omega_Z)$ represent the relative translation and rotation of a given object in the scene.

The motion, relative to the camera, of a rigid object in the scene can be decomposed into two components: translation $\mathbf{T} = (T_X, T_Y, T_Z)$ and rotation $\mathbf{\Omega} = (\Omega_X, \Omega_Y, \Omega_Z)$. In the equations corresponding to velocity fields, these symbols represent instantaneous spatial velocities, and, in the equations corresponding to displacement fields, they represent differences in position and orientation between two time instances.

In the velocity-based scheme, if (X, Y, Z) are the instantaneous camera coordinates of a point on the object, then the corresponding projection (x, y) on the image moves with a velocity (α, β) , where [LON80]:

$$\alpha = -\Omega_X xy + \Omega_Y(1 + x^2) - \Omega_Z y + (T_X - T_Z x)/Z \quad (3.2a)$$

and

$$\beta = -\Omega_X(1 + y^2) + \Omega_Y xy + \Omega_Z x + (T_Y - T_Z y)/Z. \quad (3.2b)$$

Notice that (α, β) can be represented as the sum

$$(\alpha, \beta) = (\alpha_R, \beta_R) + (\alpha_T, \beta_T), \quad (3.3)$$

where (α_R, β_R) and (α_T, β_T) are, respectively, the rotational and translational components of the velocity field:

$$\alpha_R = -\Omega_X xy + \Omega_Y(1 + x^2) - \Omega_Z y, \quad \alpha_T = (T_X - T_Z x)/Z, \quad (3.4a,b)$$

$$\beta_R = -\Omega_X(1 + y^2) + \Omega_Y xy + \Omega_Z x, \quad \beta_T = (T_Y - T_Z y)/Z. \quad (3.4c,d)$$

In the displacement-based scheme, let (X, Y, Z) be the camera coordinates at time t_1 of a point on the object and let (X', Y', Z') be the corresponding coordinates at time t_2 .

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = R \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \mathbf{T}, \quad (3.5)$$

where the rotation matrix R can be approximated, assuming small values of the rotation parameters, by:

$$R = \begin{pmatrix} 1 & -\Omega_Z & \Omega_Y \\ \Omega_Z & 1 & -\Omega_X \\ -\Omega_Y & \Omega_X & 1 \end{pmatrix}. \quad (3.6)$$

If (x, y) and (x', y') are the image coordinates corresponding to the points (X, Y, Z) and (X', Y', Z') , respectively, then:

$$x' = \frac{X'}{Z'} = \frac{x - \Omega_Z y + \Omega_Y + T_X/Z}{-\Omega_Y x + \Omega_X y + 1 + T_Z/Z} \quad (3.7a)$$

and

$$y' = \frac{Y'}{Z'} = \frac{\Omega_Z x + y - \Omega_X + T_Y/Z}{-\Omega_Y x + \Omega_X y + 1 + T_Z/Z}. \quad (3.7b)$$

Now, let (α, β) be, in this case, the displacement vector $(x' - x, y' - y)$. Then from (3.7) we get:

$$\alpha = \frac{-\Omega_X xy + \Omega_Y(1 + x^2) - \Omega_Z y + (T_X - T_Z x)/Z}{1 + \Omega_X y - \Omega_Y x + T_Z/Z} \quad (3.8a)$$

and

$$\beta = \frac{-\Omega_X(1 + y^2) + \Omega_Y xy + \Omega_Z x + (T_Y - T_Z y)/Z}{1 + \Omega_X y - \Omega_Y x + T_Z/Z}. \quad (3.8b)$$

If $|T_Z/Z| \ll 1$ and the field of view of the camera, i.e., the visual angle corresponding to the whole image, is not very large, then (employing also the assumption that the rotation parameters are small) we can approximate the displacement vector (α, β) by equations (3.2).

To conclude: equations (3.2) hold not only for velocity fields, but also for displacement fields, given that the rotation parameters are small and that the Z -component of the translation is small relative to the distance of the

object from the image plane. Such assumptions are reasonable if the time interval between the two image frames is short enough or if the motion is slow. In the following sections we restrict ourselves to conditions which allow us to employ equations (3.2) as the basis of our analysis.

3.2 The Task — Inputs and Outputs

The input utilized by our scheme for interpreting motion information is a flow field described by $\{(\alpha(x, y), \beta(x, y), W(x, y))\}$, where $(\alpha(x, y), \beta(x, y))$ is the flow vector at the (x, y) pixel in the image and $W(x, y)$ is a corresponding weight between 0 and 1. High reliability of the flow vector is represented by a weight close to 1 and low reliability by a weight close to 0. The flow field can be either dense, thus defined at most of the pixels, or sparse, thus defined only on a sparse subset of the image pixels. If the flow field is undefined at a pixel (x, y) , then $W(x, y)$ is determined to be 0. A rough estimate of the noise level in the flow field is assumed to be known.

The interpretation process should result in three outputs: object masks, motion parameters and depth. We want to partition the set $\{(x, y) : W(x, y) > 0\}$ into disjoint sets of pixels, where each set corresponds to a different rigid object. The pixels corresponding to the stationary environment, where the optical flow is induced only by the camera motion, should be grouped together.

The 5 recoverable motion parameters of each rigid object, relative to the camera, should be estimated. These parameters include the rotation parameters $(\Omega_X, \Omega_Y, \Omega_Z)$ and the direction of the translation vector defined by the unit vector $\underline{U} = \underline{T}/r$, where r is the length of the translation vector \underline{T} . Once the motion parameters are recovered, it is also possible to estimate the relative depth, $Z(x, y)/r$, corresponding to each pixel (x, y) where a flow vector is defined, unless $r = 0$ or the location of the vector is exactly in the FOE.

4. SEGMENTATION

In this section we develop a method for segmentation of the flow field into connected sets of flow vectors, where each set is consistent with a rigid motion of a roughly planar patch. A segment, satisfying this constraint, is very likely to correspond to a portion of only one rigid object. Thus, the data is organized into coherent units which form the basis for further processing. Another purpose of the segmentation is exclusion of incorrect flow vectors which are inconsistent with their neighbors.

4.1 Ψ Transformations — A Segmentation Constraint

In order to achieve a useful segmentation, we employ a few simple observations on the structure of optical flow fields. First, we examine the flow field induced by a rigid motion of a planar surface. Excluding the degenerate case in which the same plane contains both the surface and the

nodal point (and, therefore, the corresponding region in the image is a straight line), the surface can be represented by the equation

$$k_1 X + k_2 Y + k_3 Z = 1. \quad (4.1)$$

The coefficients k_1 , k_2 and k_3 can be any real numbers, except the case in which all of them are zero. Using (3.1), we obtain:

$$1/Z = k_1 x + k_2 y + k_3. \quad (4.2)$$

Substituting (4.2) in (3.2), we realize that, given a relative motion $\{\underline{T}, \underline{\Omega}\}$, the flow field is:

$$\alpha = a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 xy, \quad (4.3a)$$

$$\beta = a_4 + a_5 x + a_6 y + a_7 xy + a_8 y^2, \quad (4.3b)$$

where:

$$a_1 = \Omega_Y + k_3 T_X, \quad (4.4a)$$

$$a_2 = k_1 T_X - k_3 T_Z, \quad (4.4b)$$

$$a_3 = -\Omega_Z + k_2 T_X, \quad (4.4c)$$

$$a_4 = -\Omega_X + k_3 T_Y, \quad (4.4d)$$

$$a_5 = \Omega_Z + k_1 T_Y, \quad (4.4e)$$

$$a_6 = k_2 T_Y - k_3 T_Z, \quad (4.4f)$$

$$a_7 = \Omega_Y - k_1 T_Z, \quad (4.4g)$$

and

$$a_8 = -\Omega_X - k_2 T_Z. \quad (4.4h)$$

Note that similar results are introduced in [WAX83]. Equations (4.3) represent what we shall call a Ψ transformation. This is a 2-D transformation of the image into itself based on the 8 parameters a_1, \dots, a_8 .

We proceed now with another observation, related to arbitrary surfaces in the environment. Given such a surface, it can be described as a function $Z = Z(x, y)$ defined on the image region R which corresponds to the projection of this surface. Let $Z' = Z'(x, y)$ be an approximation to the surface Z such that

$$|\Delta Z(x, y)| \stackrel{\text{def}}{=} |Z(x, y) - Z'(x, y)| \ll Z(x, y) \quad \forall (x, y) \in R. \quad (4.5)$$

If (α_T, β_T) and (α'_T, β'_T) are the translational components of the flow fields induced by the same motion of the surfaces Z and Z' , respectively, then

$$\begin{aligned} \alpha'_T &= \frac{T_X - T_Z x}{Z'} = \frac{T_X - T_Z x}{Z - \Delta Z} \approx \frac{T_X - T_Z x}{Z} (1 + \Delta Z/Z) \\ &= \alpha_T (1 + \Delta Z/Z) \end{aligned} \quad (4.6a)$$

and

$$\begin{aligned} \beta'_T &= \frac{T_Y - T_Z y}{Z'} = \frac{T_Y - T_Z y}{Z - \Delta Z} \approx \frac{T_Y - T_Z y}{Z} (1 + \Delta Z/Z) \\ &= \beta_T (1 + \Delta Z/Z). \end{aligned} \quad (4.6b)$$

The rotational component of the flow field is independent of the structure of the environment. Hence, given (4.5), the flow field induced by the approximating surface Z' is very similar to the real flow in the region R . As a conclusion, if Z' is a planar surface which satisfies equation (4.5), then the flow field in R can be approximated by a Ψ transformation.

In a real world environment the surface can be usually approximated by a piecewise planar surface, containing only a few planar patches, where the distance between the real surface and the approximating one is small relative to the distance from the sensor to the surface. If this is the case, then the flow field can be approximated, reasonably well, by a piecewise Ψ transformation. This suggests that a useful segmentation of the flow field can be based on finding connected sets of flow vectors, where each set approximately satisfies the same Ψ transformation. Thus, each segment is consistent with a rigid motion of a roughly planar surface and can be assumed to be induced by the relative motion of only one rigid object. In the next section we describe an algorithm for achieving such a segmentation.

4.2 Segmentation Algorithm

The generalized Hough transform technique [BAL81a] is a useful tool for grouping together flow vectors which satisfy the same 2-D parameterized transformation [ADI83]. In this technique, the set of relevant transformations is represented by a discrete multi-dimensional parameter space, where each dimension corresponds to one of the transformation parameters. Each point in this space uniquely characterizes a transformation, defined by the corresponding parameter values. A flow vector 'votes' for each point with an associated transformation consistent with this vector. The points receiving the most votes are likely to represent transformations corresponding to large segments in the flow field.

As a global technique, the Hough transform is relatively insensitive to noise and partially incorrect or occluded data. However, high dimensionality of the parameter space requires large amounts of memory and computation time. In our case, the segmentation constraint is based on the 8-parameter Ψ transformations (equations (4.3)). The Hough technique can, in principle, be employed, but the computational cost required for such a number of parameters is very high. Therefore, a three-stage algorithm is proposed.

The first stage is based on grouping together adjacent flow vectors into *components* consistent with *affine transformations*. The affine transformations, represented by

$$\alpha = a_1 + a_2x + a_3y \quad (4.7a)$$

and

$$\beta = a_4 + a_5x + a_6y, \quad (4.7b)$$

are sub-class of the Ψ transformations, parameterized by only 6 parameters. Furthermore, these parameters can be partitioned into two disjoint sets of 3 parameters each, corresponding to equations (4.7a) and (4.7b). Thus, the grouping problem in the first stage can be basically solved by applying the Hough technique to 3-dimensional parameter spaces, as will be shown in sub-section 4.2.1.

In the second stage, components which are consistent with the same Ψ transformation are merged into segments. Given a set of adjacent components, optimal parameters are computed, using the least-squares technique. Related error measures, associated with each component in the set, can be then obtained. If these error values are not high (in a sense defined in sub-section 4.2.2), then the components are merged.

Sometimes over-fragmentation may occur in the first stage of the segmentation, that is, a segment is partitioned into a large number of small components, as demonstrated in experiment 1 in section 6.1 (see figure 6.1b). In order to reduce the computational cost of the first and second segmentation stages, the grouping of vectors belonging to small connected sets may be postponed, in such a case, to the third stage. In this stage, flow vectors which are not contained in any of the segments are merged into neighboring segments, if they are consistent with the corresponding Ψ transformations. If, after the third stage, some of these small sets are still not merged into the existing segments, then the first and second stages of the segmentation may be repeated, focused only on these sets, thus possibly creating new segments. In the following sub-sections the three stages of the segmentation are more fully described, but, for the sake of brevity, many details are still suppressed.

4.2.1 First Stage — Grouping Based on Affine Transformations

4.2.1.1 A modified version of the generalized Hough technique

The grouping of flow vectors into components consistent with affine transformations is based on a modification of the generalized Hough technique. The affine transformations can be represented by a 6-dimensional parameter space where each dimension corresponds to one of the parameters a_1, \dots, a_6 in equations (4.7). For computational reasons the parameter space should contain only a finite number of points. Therefore, minimal and maximal values are determined for each parameter and the corresponding interval is quantized. The parameter space is the cartesian product of the obtained sets.

A flow vector $(\alpha(x, y), \beta(x, y))$ votes for a transformation (a_1, \dots, a_6) , if it approximately satisfies the constraint equations (4.7), that is, if

$$\delta \stackrel{\text{def}}{=} \sqrt{\delta_x^2 + \delta_y^2} \leq \epsilon, \quad (4.8a)$$

where

$$\delta_x = |\alpha - a_1 - a_2x - a_3y| \quad (4.8b)$$

and

$$\delta_y = |\beta - a_4 - a_5x - a_6y|. \quad (4.8c)$$

Note that ϵ is a function of the resolution in the parameter space and the noise level in the flow field, but it is never less than a given threshold, typically one pixel. In this case, the amount of support is determined by the function

$$V(a_1, a_2, a_3, a_4, a_5, a_6, x, y) = 1 - 0.75\delta/\epsilon \quad (4.9)$$

which allows the support to range from 1 down to 0.25 for those flow vectors at the limit of the acceptable error range. The total amount of support, given to each transformation (a_1, \dots, a_6) , is the weighted sum

$$S(a_1, a_2, a_3, a_4, a_5, a_6) = \sum_{x,y} W(x,y) V(a_1, a_2, a_3, a_4, a_5, a_6, x, y), \quad (4.10)$$

where $W(x, y)$ is the weight of the flow vector in the pixel (x, y) .

Suppose now that we want to find the affine transformation, among those represented in the parameter space, which is maximally supported by a given set of flow vectors. Basically, we have to compute the support, according to equation (4.10), given to any of those transformations. A serious computational problem may arise if the number of points in the parameter space is very high. If, for example, the minimal and maximal possible values of the parameter a_1 are -64 pixels and +64 pixels, respectively, and the desired accuracy is 0.25 pixel, then 512 samples are apparently needed for this parameter. If an equal number of samples is also required for the other parameters, then the parameter space should contain $512^6 \approx 16 \times 10^{15}$ points. In such a case, a straightforward Hough technique is computationally impractical.

This problem is alleviated by using two techniques. First, a multi-resolution scheme in the parameter space is employed. The Hough technique is iteratively used, where in each iteration the parameter space is quantized around the values estimated in the previous iteration, using a finer resolution. Thus, utilizing a limited memory size, accurate parameter values can still be found. Other methods for achieving this goal are presented in [ORO81, SLO81].

The second technique is based on decomposition of the parameter set into two disjoint subsets, $\{a_1, a_2, a_3\}$ and $\{a_4, a_5, a_6\}$. The Hough technique is separately applied to the corresponding 3-dimensional parameter spaces, using the relevant constraint, (4.7a) or (4.7b). Sets of highly supported parameter triples, $A_\alpha = \{(a_{1i}, a_{2i}, a_{3i}) : i = 1, \dots, N\}$ and $A_\beta = \{(a_{4i}, a_{5i}, a_{6i}) : i = 1, \dots, N\}$, are thus found, where N was experimentally determined to be 10. As a result, a set of N^2 hypothesized affine transformations,

$$A_{\alpha\beta} = A_\alpha \times A_\beta = \{(a_{1i}, a_{2i}, a_{3i}, a_{4j}, a_{5j}, a_{6j}) : i, j = 1, \dots, N\}, \quad (4.11)$$

is obtained. The support function can be then directly applied to the set $A_{\alpha\beta}$, thus determining the maximally supported transformation T^* in this set. T^* is not necessarily the maximally supported transformation in the 6-dimensional parameter space. However, large components in the flow field, corresponding to maxima points in the 6-dimensional space, can be expected to produce maxima points also in each of the 3-dimensional parameter spaces. Therefore T^* is, at least, a near optimal transformation, as can also be concluded from the experimental results. The decomposition technique is employed in each iteration of the multi-resolution scheme; together they create a very efficient algorithm.

4.2.1.2 Implementation of a multipass approach

The components which we try to locate are connected sets of flow vectors which support the same affine transformation. The algorithm for obtaining this goal is based on a multipass Hough approach, where a basic cycle of operations is repeatedly executed [FEN79, ADI83]. The input to each cycle includes masks of the components which were already detected during the previous cycles and a mask of those vectors which were excluded from further consideration. The cycle is composed of the following steps:

- 1) Consider the set of pixels which are assigned a positive weight, do not belong to any of the previously found components and were not excluded from further consideration. Find in this set a connected subset E with maximal sum of weights. If this sum is below a given threshold L , which is related to the noise level in the flow field, then stop searching for new components and start the merging stage. Sometimes over-fragmentation occurs, i.e., a segment is partitioned into a large number of small components. In order to prevent an excessive number of cycles in such a case, a new threshold, higher than L , is determined and the process is stopped if the sum of weights is below this threshold. The grouping of vectors in small sets is thus postponed to the third stage.

- 2) Partition the set E into a given number (typically 64) of square windows, such that the sum of weights in each window is roughly the same. Then, from each window, select the flow vector with maximal weight. The Hough technique will be applied only to these vectors, and not to the whole set E , in order to reduce the computation time.

- 3) Use the modified Hough technique, described in section 4.2.1.1, to find the affine transformation which receives the maximal support from the flow vectors selected in the previous step.

- 4) Determine the set F of all the vectors in E which are consistent with the computed affine transformation. If the sum of weights corresponding to F is below the threshold L , then exclude the set E from further consideration and

start a new cycle. Otherwise, find in F a connected subset G with maximal sum of weights. Then, if this sum exceeds L , add G to the list of components; otherwise, just exclude G from further consideration (to prevent an infinite loop).

4.2.2 Second Stage — Merging of Components

Components, created in the first stage of the segmentation, are atomic units which, if consistent with the same Ψ transformation, should be merged together to create a segment. Two main steps can be observed in the merging process. In the first step an optimal Ψ transformation is estimated for each component, employing the least squares technique. If the component contains n flow vectors, then the error function to be minimized is

$$E(a_1, \dots, a_8) = \sum_{i=1}^n W_i \left[(\alpha_i - a_1 - a_2 x_i - a_3 y_i - a_7 x_i^2 - a_8 x_i y_i)^2 + (\beta_i - a_4 - a_5 x_i - a_6 y_i - a_7 x_i y_i - a_8 y_i^2)^2 \right], \quad (4.12)$$

where, for each $1 \leq i \leq n$, $(\alpha_i, \beta_i) = (\alpha(x_i, y_i), \beta(x_i, y_i))$ is a flow vector and W_i is the corresponding weight. Taking partial derivatives with respect to a_1, \dots, a_8 and equating to 0, a set of linear equations is obtained. Their solution, a_1^*, \dots, a_8^* , is the optimal Ψ transformation. Substituting this solution in (4.12) and using the normalization equation

$$\sigma = \sqrt{E(a_1^*, \dots, a_8^*) / \sum_{i=1}^n W_i}, \quad (4.13)$$

an error value, corresponding to the component, is obtained. σ is an estimate of the standard deviation of the actual flow values from those predicted by the optimal Ψ transformation.

In the second step Ψ transformations, corresponding to merged sets of adjacent components, are computed. Based on related error values, associated with each component in such a set, it is decided whether to merge the components. In order to formulate the conditions for a merging decision, some notations are used. First, S denotes a set of adjacent components and Ψ_S is the corresponding optimal Ψ transformation. For each component C_j in S , σ_j is the error value found in the first step of the merging process, σ'_j is the error value obtained by substituting the coefficients of Ψ_S in (4.13), and p_j is the ratio between the sum of vector weights in C_j and the total sum of weights in the set S .

The ratios $\{\sigma'_j/\sigma_j\}$ are a major factor in the merging decision. If these ratios are only slightly higher than 1, then a merging decision seems to be justified. Note that σ'_j is never less than σ_j , because the optimal Ψ transformation corresponding to the component C_j can be adjusted to the local surface and noise associated with C_j . If, however, p_j is close to 1, then we expect σ'_j to be very close to σ_j ,

especially when a merging decision is justified. Hence, the allowed level of σ'_j/σ_j will be defined to be a monotonically decreasing function of p_j :

$$L_a(p_j) = \tau_1 - (\tau_1 - 1)p_j, \quad (4.14)$$

where τ_1 is a given threshold (typically $\tau_1 \approx 1.5$). Thus, $L_a(p_j)$ ranges from values close to 1 for components with relatively large weight, up to almost τ_1 for small components.

Sometimes, however, σ_j can not be computed because the linear equations derived from (4.12) are linearly dependent. In addition, if the component C_j is small, then σ_j may be unreliable as a basis for evaluation of σ'_j values. Therefore, an absolute threshold $L_b(p_j)$ of allowed values of σ'_j is also employed. $L_b(p_j)$ is given by

$$L_b(p_j) = \tau_3 - (\tau_3 - \tau_2)p_j, \quad (4.15)$$

where τ_2 and τ_3 are pre-determined thresholds related to the expected noise level in the flow field and $\tau_3 > \tau_2$. Thus $L_b(p_j)$ ranges from τ_2 for very large components up to τ_3 for small components. The reason for this dependency on p_j is related to the effect of statistical averaging of the noise. In large objects, such averaging is likely to take place and thus τ_2 represents the estimated standard deviation of the noise. The threshold τ_3 , on the other hand, represents some reasonable upper bound of the noise level. If, for example, the most significant noise is induced by using flow values rounded to integers and, therefore, the noise is uniformly distributed between -0.5 pixels and $+0.5$ pixels, then τ_2 will be taken to be the corresponding standard deviation, that is, approximately 0.3 pixels and τ_3 will be 0.5 pixels. To conclude, a merging decision is accepted if and only if, for each component C_j in S , $\sigma'_j/\sigma_j \leq L_a(p_j)$ or $\sigma'_j \leq L_b(p_j)$, i.e.,

$$\sigma'_j \leq \max\{L_a(p_j)\sigma_j, L_b(p_j)\}. \quad (4.16)$$

The algorithm, for finding sets of components to be merged, starts with detection of the component with the maximal sum of vector weights. Then, merging of this component with its neighbors is sequentially tested, in the order of their associated sums of weights. If one of these merging trials is successful, then merging of additional components with the already merged pair is examined. In general, given a set of already merged components, neighboring components are tested as candidates for adjoining this set. This process continues until all the candidates for merging are examined. Then, the process starts again, considering only the components which are not yet assigned to any of the already created segments. Eventually, all the components are contained in one of the segments.

4.2.3 Third Stage

The purpose of the third stage of the segmentation is examination of flow vectors which were assigned positive weights and were not grouped into any of the components in the first stage of the segmentation, and thus do not belong to any of the segments. Such vectors, called 0-vectors, which are neighbors of one of the segments, are tested for consistency with the Ψ transformation corresponding to this segment and, if consistent, are merged into it. Then, 0-vectors, neighbors of the just segmented vectors, are examined in their turn. This process is iteratively executed until no new vector is merged into one of the segments.

It is possible that after the third stage, connected sets of 0-vectors, which were not excluded from further consideration in the first segmentation stage, are still not contained in any of the existing segments. In such a case, the first and the second stages of the segmentation are executed again, focused only on these sets, thus possibly creating new segments.

5. FORMING OBJECT HYPOTHESES AND RECOVERING 3-D INFORMATION

In the first stage of the interpretation process, described in the preceding section, the flow field is segmented into connected sets of flow vectors, where each set is consistent with a rigid motion of a roughly planar surface. Such a segment is assumed to correspond to a portion of only one rigid object. The next task is to detect sets of segments which are consistent with the same 3-D motion parameters. Such a set can be hypothesized, employing the rigidity assumption [ULL79], to be induced by one rigidly moving object (or by the camera motion). In sub-section 5.1 we describe an algorithm for computing the motion parameters from a set of flow vectors generated by a rigid motion. In section 5.2 we combine this algorithm with the segmentation results to form object hypotheses and estimate the corresponding 3-D motion and structure. Again, for the sake of brevity, many details are suppressed.

5.1 Estimating Motion Parameters and Depth Information of a Rigid Object

5.1.1 Optimisation Constraint

Given a set of flow vectors, assumed to be induced by a rigidly moving object, we want to find the 3-D motion parameters which are maximally consistent with this data. Following [BRU81], we employ the least-squares approach because of its relative robustness in the presence of noise. Based on (3.2), the error function to be minimized is

$$\sum_{i=1}^n W_i \left[\left(\alpha_i + \Omega_X x_i y_i - \Omega_Y (1 + x_i^2) + \Omega_Z y_i - \frac{T_X - T_Z x_i}{Z_i} \right)^2 + \left(\beta_i + \Omega_X (1 + y_i^2) - \Omega_Y x_i y_i - \Omega_Z x_i - \frac{T_Y - T_Z y_i}{Z_i} \right)^2 \right], \quad (5.1)$$

where $\mathbf{T} = (T_X, T_Y, T_Z)$ and $\mathbf{\Omega} = (\Omega_X, \Omega_Y, \Omega_Z)$ are the translation and rotation vectors, respectively, and, for each i between 1 and n , (α_i, β_i) is the flow vector computed at the pixel (x_i, y_i) , W_i is its weight and Z_i is the spatial depth of the corresponding point in the environment. The task is to determine \mathbf{T} , $\mathbf{\Omega}$ and $\{Z_i\}$ which minimize this function. Using the decomposition of the flow field into its rotational and translational components, denoted by (α_R, β_R) and (α_T, β_T) (see equations (3.4)), the error function can be more concisely represented by

$$\sum_{i=1}^n W_i [(\alpha_i - \alpha_R - \alpha_T)^2 + (\beta_i - \beta_R - \beta_T)^2]. \quad (5.2)$$

As can be easily seen, it is actually impossible to determine the absolute values of (T_X, T_Y, T_Z) and $\{Z_i : i = 1, \dots, n\}$. However, if the length, denoted by r , of the translation vector is non-zero, then it is possible to estimate the direction of the 3-D translation, represented by the unit vector

$$(U_X, U_Y, U_Z) = (\mathbf{T}_X, \mathbf{T}_Y, \mathbf{T}_Z)/r, \quad (5.3)$$

and the relative depth values, represented by

$$\bar{Z}_i = r/Z_i, \quad i = 1, \dots, n. \quad (5.4)$$

Introducing the abbreviations

$$\alpha_U = U_X - U_Z x = \alpha_T / \bar{Z} \quad (5.5a)$$

and

$$\beta_U = U_Y - U_Z y = \beta_T / \bar{Z}, \quad (5.5b)$$

(5.2) can be rewritten as

$$\sum_{i=1}^n W_i [(\alpha_i - \alpha_R - \alpha_U \bar{Z}_i)^2 + (\beta_i - \beta_R - \beta_U \bar{Z}_i)^2]. \quad (5.6)$$

Thus, the task can be reformulated as finding the values of $(\Omega_X, \Omega_Y, \Omega_Z)$, (U_X, U_Y, U_Z) and $\{\bar{Z}_i : i = 1, \dots, n\}$ which minimize this expression. In addition, the depth constraints

$$\bar{Z}_i \geq 0, \quad i = 1, \dots, n, \quad (5.7)$$

should be satisfied. Note that this error measure is different from the one employed in [BRU81] where the contribution of each flow vector is multiplied by $\alpha_U^2 + \beta_U^2$.

For any given i , $1 \leq i \leq n$, we can find the optimal value of \bar{Z}_i , as a function of the motion parameters, by examining the first derivative of (5.6) with respect to \bar{Z}_i . This derivative is given by

$$2W_i [-(\alpha_i - \alpha_R)\alpha_U - (\beta_i - \beta_R)\beta_U + (\alpha_U^2 + \beta_U^2)\bar{Z}_i]. \quad (5.8)$$

Setting it equal to 0 yields

$$\tilde{Z}_i = ((\alpha_i - \alpha_{Ri})\alpha_{Vi} + (\beta_i - \beta_{Ri})\beta_{Vi}) / (\alpha_{Vi}^2 + \beta_{Vi}^2), \quad (5.9)$$

unless $\alpha_{Vi}^2 + \beta_{Vi}^2 = 0$, in which case \tilde{Z}_i can be assigned any non-negative value. If the expression in (5.9) is negative, then the corresponding depth constraint in (5.7) is unsatisfied. In such a case, to minimize the error function (5.6), \tilde{Z}_i should be set to 0, since the derivative (5.8) is non-negative for non-negative values of \tilde{Z}_i and, therefore, the error function is monotonically non-decreasing for these values. To summarize, the optimal value of \tilde{Z}_i is given by

$$\tilde{Z}_i = [\delta_i / (\alpha_{Vi}^2 + \beta_{Vi}^2)]^+, \quad (5.10)$$

where $\delta_i = (\alpha_i - \alpha_{Ri})\alpha_{Vi} + (\beta_i - \beta_{Ri})\beta_{Vi}$. Substituting (5.10), for any $1 \leq i \leq n$, into (5.6) and expanding the resulting expression yields the following representation of the error, as a function of the motion parameters:

$$E(\underline{U}, \underline{\Omega}) = \sum_{i=1}^n W_i E_i, \quad (5.11a)$$

where

$$E_i = \begin{cases} \frac{[(\alpha_i - \alpha_{Ri})\beta_{Vi} - (\beta_i - \beta_{Ri})\alpha_{Vi}]^2}{\alpha_{Vi}^2 + \beta_{Vi}^2} & \text{if } \delta_i > 0; \\ (\alpha_i - \alpha_{Ri})^2 + (\beta_i - \beta_{Ri})^2 & \text{otherwise.} \end{cases} \quad (5.11b)$$

A normalized version of this error function, defined by

$$\sigma(\underline{U}, \underline{\Omega}) = \sqrt{E(\underline{U}, \underline{\Omega}) / \sum_{i=1}^n W_i}, \quad (5.12)$$

will be also utilized. σ is an estimate of the standard deviation of the measured flow values from those predicted by the motion parameters and the corresponding depth values.

Note that the expression (5.11) for the error function was obtained by assuming non-zero translation. In the case of a purely rotational motion, the appropriate error function to be minimized is:

$$E_R(\underline{\Omega}) = \sum_{i=1}^n W_i ((\alpha_i - \alpha_{Ri})^2 + (\beta_i - \beta_{Ri})^2). \quad (5.13)$$

If the minimal value of this function is close to the minimal value of the function (5.11), then the motion is, possibly, purely rotational.

The task of finding the rotation parameters which minimize the function $E_R(\underline{\Omega})$ can be easily accomplished by solving a set of three linear equations [BRU81] and will not

be considered further in this paper. Thus, in the next section, we concentrate on the much more difficult task of finding values of \underline{U} and $\underline{\Omega}$ which minimize the error function $E(\underline{U}, \underline{\Omega})$ (or, equivalently, the function $\sigma(\underline{U}, \underline{\Omega})$), where \underline{U} can be any unit vector and $\underline{\Omega}$ is unconstrained.

5.1.2 Algorithm

The algorithm for recovering the motion parameters employs an error measure, derived from (5.12), corresponding to possible directions of the translation vector. A minimum value of this function is determined, using a multi-resolution sampling scheme.

Let us start the derivation of this error measure with the observation that if the depth constraints (5.7) are ignored, then, for any hypothesized direction of translation, the optimal rotation parameters can be easily extracted by solving a set of three linear equations. To see that, notice that the error function (5.11) can be reduced in this case to the function

$$E'(\underline{U}, \underline{\Omega}) = \sum_{i=1}^n W_i \frac{((\alpha_i - \alpha_{Ri})\beta_{Vi} - (\beta_i - \beta_{Ri})\alpha_{Vi})^2}{\alpha_{Vi}^2 + \beta_{Vi}^2}. \quad (5.14)$$

Differentiating $E'(\underline{U}, \underline{\Omega})$ with respect to the rotation parameters and setting the derivatives equal to 0 yields three linear equations with the rotation parameters as unknowns. Thus, ignoring the depth constraints (5.7), the search space can be limited to the unit sphere $\{\underline{U} : |\underline{U}| = 1\}$.

Moreover, changing the sign of any unit vector \underline{U} has no effect on the value of $E'(\underline{U}, \underline{\Omega})$ since it only affects the sign of α_{Vi} and β_{Vi} . Therefore, the search space can be further restricted to the hemisphere

$$HS = \{\underline{U} : |\underline{U}| = 1 \text{ and } U_z \geq 0\}. \quad (5.15)$$

The preferred sign of \underline{U} can be then determined, as proposed in [BRU81], as the one which gives $\tilde{Z}_i \geq 0$ for most indices i . Still, we wish to incorporate these constraints or, equivalently, the equations (5.11b) in a more rigorous way. Hence, for each \underline{U} in HS , we define the error measure

$$\sigma_1(\underline{U}) = \min_{\underline{B}, \underline{\Omega}} \sigma(B\underline{U}, \underline{\Omega}), \quad (5.16)$$

where B can have the values $+1$ or -1 . The goal is to find a vector \underline{U} in HS which minimizes the function σ_1 . The associated values of B and $\underline{\Omega}$ are, respectively, the determined sign of the translation vector and the estimated rotation parameters. The function σ_1 is, however, difficult to compute. Therefore, in the proposed algorithm we compute an approximation to σ_1 which is experimentally shown to be very accurate. A few main steps can be distinguished in the procedure for computing this approximation:

1) Given a vector \underline{U} in HS , estimate the optimal rotation vector $\underline{\Omega}^*$ by minimizing $E'(\underline{U}, \underline{\Omega})$ with respect to

$\underline{\Omega}$, and compute the corresponding normalized error measure $\sigma'(\underline{U}, \underline{\Omega}^*)$. This error value is a lower bound of $\sigma_1(\underline{U})$ since it minimizes the error function $\sigma(\underline{U}, \underline{\Omega})$, with respect to $\underline{\Omega}$ and the sign of \underline{U} , without considering the depth constraints (5.7).

2) Compute $\sigma(\underline{U}, \underline{\Omega}^*)$ and $\sigma(-\underline{U}, \underline{\Omega}^*)$. Determining the minimum of these two error values yields the preferred sign, denoted by μ , of \underline{U} . Using the notation $\underline{U}^* = \mu \underline{U}$, $\sigma(\underline{U}^*, \underline{\Omega}^*)$ is an upper bound of $\sigma_1(\underline{U})$, because it gives the actual error measure for some values of B and $\underline{\Omega}$ in equation (5.16).

3) Compute an approximation to $\sigma_1(\underline{U})$ by averaging its lower and upper bounds:

$$\hat{\sigma}_1(\underline{U}) = (\sigma'(\underline{U}, \underline{\Omega}^*) + \sigma(\underline{U}^*, \underline{\Omega}^*)) / 2. \quad (5.17)$$

The relative deviation of $\hat{\sigma}_1(\underline{U})$ from $\sigma_1(\underline{U})$ is bounded by

$$(\sigma(\underline{U}^*, \underline{\Omega}^*) - \sigma'(\underline{U}, \underline{\Omega}^*)) / (2\hat{\sigma}_1(\underline{U})). \quad (5.18)$$

In the experiments, this value was found to be very small, typically much less than 0.01.

The search for an optimal vector in HS consists of a sampling (similar to [LAW82]) of the error measure $\hat{\sigma}_1$. A multi-resolution scheme is employed, where in the first iteration the set HS is coarsely sampled and, in each additional iteration, only the neighborhood of the vector giving a minimum value in the previous iteration is sampled, using a finer resolution. Note that solutions near the boundary of HS require a vector \underline{U}' to be defined as a 'neighbor' of a vector \underline{U} if either \underline{U}' or $-\underline{U}'$ is close to \underline{U} . Another way to obtain the same effect while using the normal definition of a neighborhood is to extend the domain of definition of the function $\hat{\sigma}_1$ to the whole unit sphere, employing exactly the same definition used for the domain HS . In this case, $\hat{\sigma}_1(-\underline{U}) = \hat{\sigma}_1(\underline{U})$ for each unit vector \underline{U} , thus, computationally, it makes no difference which domain of definition is used.

The final solution of \underline{U} , and the corresponding sign μ and the rotation parameters $\underline{\Omega}^*$, defined in the procedure for computing $\hat{\sigma}_1$, are the determined motion parameters. Substituting these parameters in equations (5.10), the relative depth, corresponding to each flow vector, can be estimated as well.

We should mention that sometimes the error function $\hat{\sigma}_1$ is very close to its minimal value in a large portion of the search space (see figure 6.2e). Hence, in the presence of noise, it may be impossible to obtain reasonably accurate estimates of the motion parameters. Two complementary approaches may be taken in order to deal with this ambiguity. First, constraints on the motion parameters and the environmental depth, rather than values, can be still recovered, using, for example, the coefficients of the related Ψ transformations (see equations (4.4)). Second, possible values of the motion parameters can be represented by a

probabilistic distribution function. Such a function can be defined, for example, on the set HS , using the computed values of $\hat{\sigma}_1$. Investigation of situations which may lead to this ambiguity is under way.

5.2 Forming Object Hypotheses

Segments of the flow field, which are consistent with the same motion parameters, can be hypothesized, using the rigidity assumption [ULL79], to be induced by one rigidly moving object (or by the camera motion). The process for detecting such sets of segments is similar to the second stage of the segmentation process, where components are merged into segments. Optimal motion parameters and a related error measure M_i are computed for each segment SEG_i , using the algorithm described in the previous section. In addition, given any set of segments, the algorithm is applied to this set and the corresponding motion parameters are computed. Then, for each segment SEG_i in the set, an error measure M'_i is obtained by substituting these parameters and the related flow data in equation (5.17). Based on the error values $\{M_i\}$ and $\{M'_i\}$, consistency of the set with rigid motion is determined, employing a decision procedure similar to the one described in section 4.2.2.

Actually, each segment is sampled, using the method in step (2) of the multipass Hough technique (section 4.2.1.2), and only the selected vectors are used for forming object hypotheses and computing the corresponding motion parameters. This sampling procedure considerably reduces the computation time. Notice that, because each segment is sampled, all the distinct surfaces and independently moving objects, even the small ones, are appropriately represented, thus preventing the suppression of valuable data.

In addition to the ambiguity described in the previous section, another ambiguity may exist in the decomposition of the environment into independently moving objects. For example, two independently moving objects induce, in some cases, a flow field which can be interpreted as resulting from one rigidly moving object. In order to deal with this ambiguity, one may have to find a set of possible decompositions, not only one. Analysis of this ambiguity is also under way.

6. EXPERIMENTS

In this section we present four experiments which demonstrate our proposed scheme for the interpretation of optical flow fields. The first two experiments are based on simulated data, and the last two are based on real data. In all the experiments, values to appear in translation vectors and surface equations are given in focal units, whereas rotation parameters are given in radians and flow vectors are given in pixel units. Actually, the flow values in the experiments based on simulated data are rounded to integers, thus inducing noise uniformly distributed between $-1/2$ and $+1/2$ pixels. The methods employed for computing the real data in experiments 3 and 4 also produce flow values given in integer units, hence the noise level

in these experiments should be at least as high as in experiments 1 and 2 (actually it is higher). The image, in all the experiments, contains 128×128 pixels. The field of view of the camera is 45° in the experiments with simulated data and 30° in the experiments with real data.

6.1 Experiment 1

The first experiment simulates a translatory motion of the camera, represented by the vectors $\underline{T}_C = (0., 0.02, 1.)$ and $\underline{\Omega}_C = (0., 0., 0.)$. The environment consists of two distinct surfaces: a plane described by the equation $Z = 50Y + 100$ and an ellipsoid represented by $(X - 2)^2 + [(Y - 2)/4]^2 + (Z - 5)^2 = 1$. A flow vector is computed for each pixel, unless the corresponding ray of light does not intersect any of the surfaces, in which case the related weight is assumed to be 0 (otherwise it is 1). A sample of the flow field is shown in figure 6.1a.

The results of the three stages of the segmentation, shown in figures 6.1b, 6.1c and 6.1d, demonstrate the role and importance of each of these stages. The two segments, found in this process, were determined to be consistent with the same rigid motion. The error function δ_1 (equation 5.17) was computed using 64 vectors from each segment. Employing a spherical coordinate system (r, ϕ, θ) , where

$$X = r \sin(\phi) \cos(\theta), \quad (6.1a)$$

$$Y = r \sin(\phi) \sin(\theta) \quad (6.1b)$$

and

$$Z = r \cos(\phi), \quad (6.1c)$$

the domain of definition of δ_1 , that is, the hemisphere $\{\underline{U} : |\underline{U}| = 1, U_z \geq 0\}$, can be represented by the set

$$\{(\phi, \theta) : 0 \leq \phi \leq 90^\circ, 0^\circ \leq \theta < 360^\circ\}. \quad (6.2)$$

This representation is utilized for displaying the function δ_1 in figure 6.1e, where (ϕ, θ) are used as polar coordinates. Employing the sampling procedure for minimizing δ_1 , the motion parameters were determined, after two iterations, to be $\underline{U} = (0.0017, -0.0204, -0.9998)$ and $\underline{\Omega} = (-0.0004, -0.0003, -0.0004)$. Note that, assuming a stationary environment, the camera motion is given by $-\underline{U}$ and $-\underline{\Omega}$. These results are in a good agreement with the correct values. Substituting the computed values in equation (5.10), the 'reciprocal depth' map, that is, the function r/Z shown in figure 6.1f, was obtained.

6.2 Experiment 2

In the second experiment, the camera motion is composed of both translation and rotation, described by $\underline{T}_C = (0.5, 0.5, 1.)$ and $\underline{\Omega}_C = (0.02, -0.02, 0.05)$. The environment contains an independently moving sphere, defined by $(X - 9)^2 + (Y - 9)^2 + (Z - 30)^2 = 4$. An object coordinate system is defined, which is parallel to the camera coordinate system but its origin is in the sphere center $(9, 9, 30)$. The motion of the object, in this coordinate system, is

represented by $\underline{T}_O = (0.5, -0.5, 0.)$ and $\underline{\Omega}_O = (0., 0., -0.2)$. The stationary environment is composed of two surfaces: a plane described by $Z = X + 0.5Y + 50$ and an ellipsoid described by $[(X + 3)/2]^2 + [(Y + 1)/5]^2 + [(Z - 20)/2]^2 = 1$. A 32×32 sample of the flow field corresponding to this scene is shown in figure 6.2a.

The segments found in the experiment are shown in figure 6.2b. The two segments associated with the stationary environment were determined to be consistent with the same rigid motion, while no rigid motion compatible with the third segment was also found to be consistent with one of the other segments. Thus, the decomposition of the flow field into sets corresponding to independently moving objects could be uniquely (and correctly) determined. The error function δ_1 corresponding to the stationary environment is displayed in figure 6.2c. The associated motion parameters of the camera were determined to be $-\underline{U} = (0.3897, 0.4017, 0.8287)$ (the corresponding actual values were $\underline{U}_C = (0.4082, 0.4082, 0.8164)$) and $-\underline{\Omega} = (0.0204, -0.0196, 0.0494)$. The related depth map is represented by the function r/Z in figure 6.2d.

The error function corresponding to the independently moving object is shown in figure 6.2e. This function is very close to its minimal value in a large portion of the search space, thus, demonstrating the ambiguity discussed in section 5.1.2.

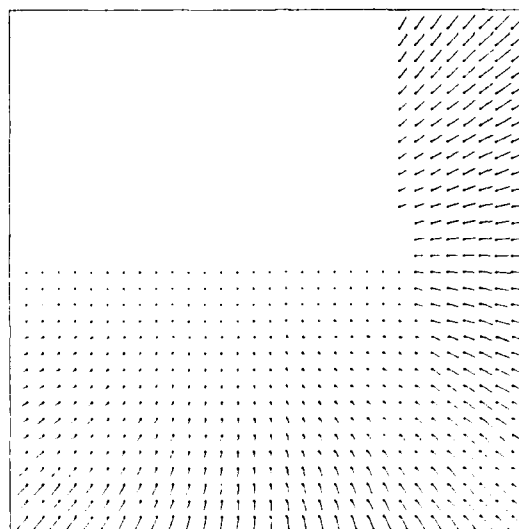


Figure 6.1: Experiment 1. (a) A sample of the flow field.

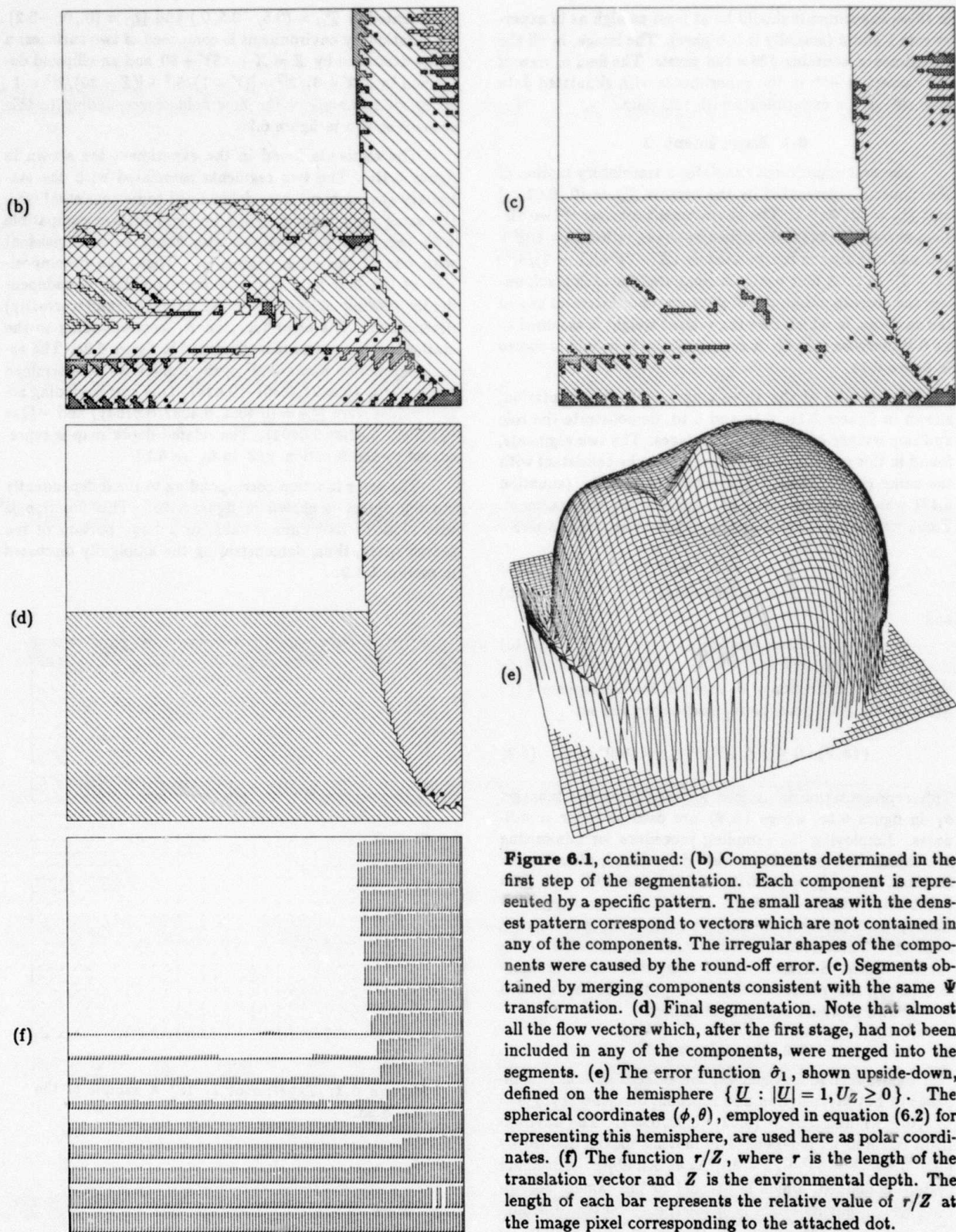


Figure 6.1, continued: (b) Components determined in the first step of the segmentation. Each component is represented by a specific pattern. The small areas with the densest pattern correspond to vectors which are not contained in any of the components. The irregular shapes of the components were caused by the round-off error. (c) Segments obtained by merging components consistent with the same Ψ transformation. (d) Final segmentation. Note that almost all the flow vectors which, after the first stage, had not been included in any of the components, were merged into the segments. (e) The error function δ_1 , shown upside-down, defined on the hemisphere $\{\underline{U} : |\underline{U}| = 1, U_z \geq 0\}$. The spherical coordinates (ϕ, θ) , employed in equation (6.2) for representing this hemisphere, are used here as polar coordinates. (f) The function r/Z , where r is the length of the translation vector and Z is the environmental depth. The length of each bar represents the relative value of r/Z at the image pixel corresponding to the attached dot.

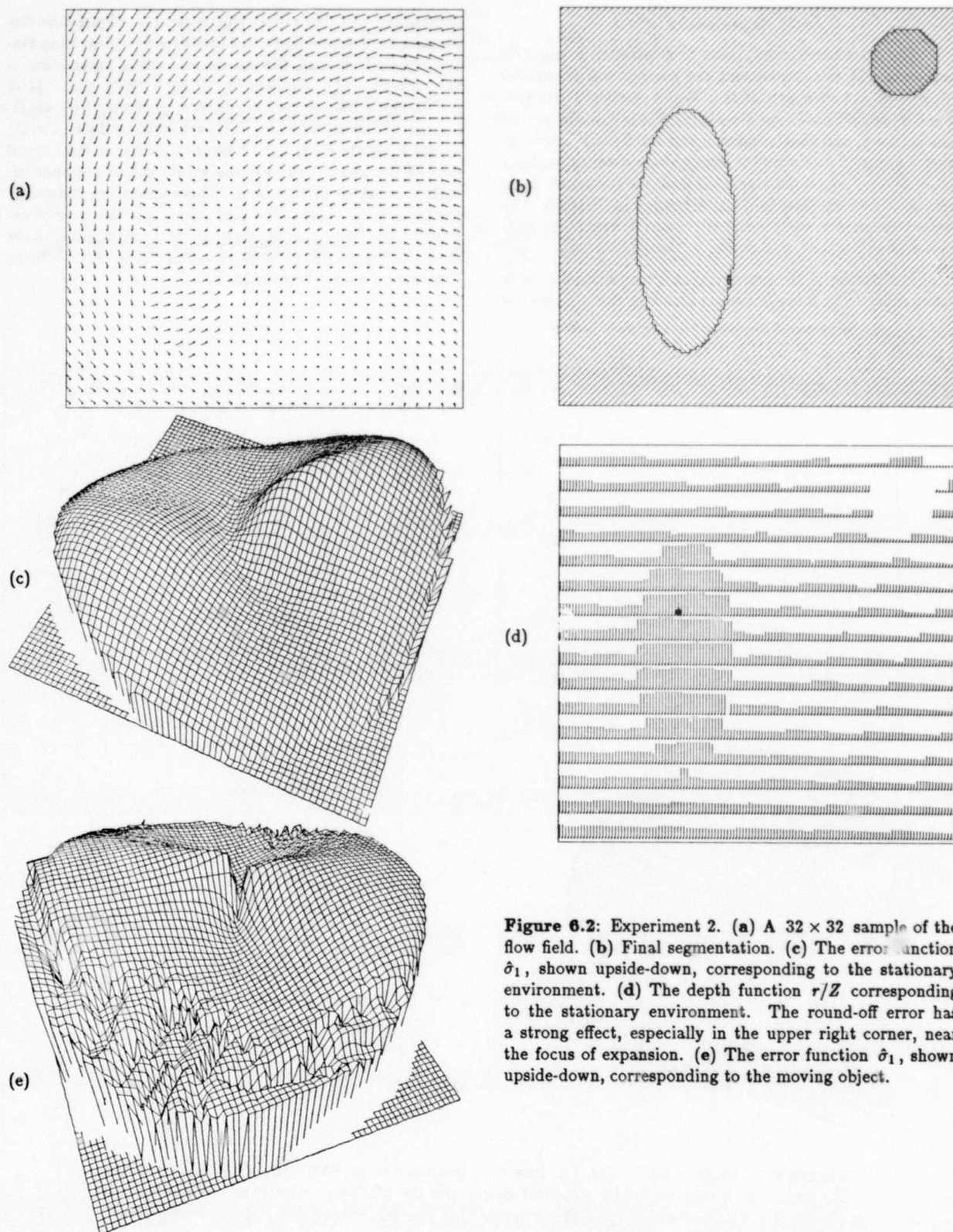


Figure 6.2: Experiment 2. (a) A 32×32 sample of the flow field. (b) Final segmentation. (c) The error function $\hat{\sigma}_1$, shown upside-down, corresponding to the stationary environment. (d) The depth function r/Z corresponding to the stationary environment. The round-off error has a strong effect, especially in the upper right corner, near the focus of expansion. (e) The error function $\hat{\sigma}_1$, shown upside-down, corresponding to the moving object.

6.3 Experiment 3

The third experiment, taken from [RIE83], is based on real data. In this experiment the camera was translated roughly in the direction of the Z -axis, between two textured cylinders, towards a textured plane parallel to the image plane, and then rotated about its Y -axis a few degrees. Figure 6.3a shows the flow vectors determined for a set of interesting points extracted from the image; for more details on how the flow field was extracted see [RIE83]. The weight assigned to each vector is 1, since no reliability measure was computed.

The results of the segmentation process are shown in figure 6.3b. The three segments found in this process are

compatible with the same camera motion. Figure 6.3c displays the corresponding error function $\hat{\sigma}_1$. Assuming stationary environment, the recovered motion parameters of the camera, $-\underline{U} = (-0.0079, 0.0181, 0.9998)$ and $-\underline{\Omega} = (-0.0018, 0.0203, -0.0006)$, are consistent with the specifications of the experiment. The estimated values of r/Z , shown in figure 6.3d, have a large variation in the central part of the image, whereas the actual values are approximately constant in this area. These errors are caused by the presence of noise in the flow values near the focus of expansion and are unavoidable in such circumstances. Note that this experiment demonstrates the ability of our scheme to interpret sparse flow fields.

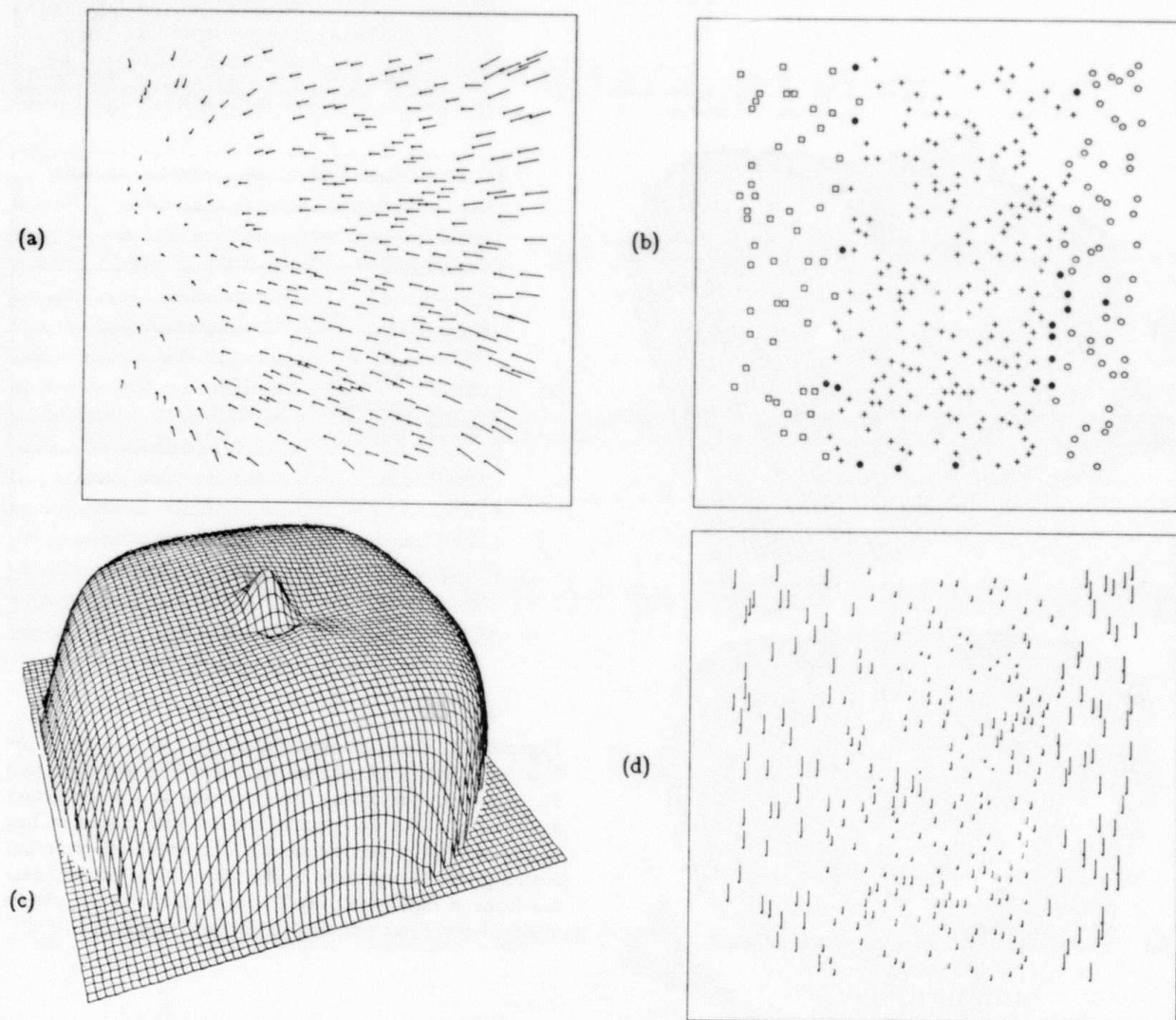


Figure 6.3: Experiment 3. (a) The flow field produced in [RIE83]. (b) Final segmentation. Each segment is represented by a distinct shape, and the black dots correspond to flow vectors which are not contained in any of the segments. (c) The error function $\hat{\sigma}_1$ shown upside-down. (d) The estimated depth function r/Z .

6.4 Experiment 4

Figures 6.4a and 6.4b are images taken from a camera translated in the direction of its X -axis. The scene mainly contains a coffee can in the front and a plant in the background. The flow field, shown in figure 6.4c, was computed using a modified version of the algorithm proposed in [GLA83]. The new version, as well as a reliability measure assigned to each flow vector, was developed by Anandan [ANA84]. Based on this reliability measure, a weight plane, shown in figure 6.4d, was computed.

The four segments in figure 6.4e were determined to be compatible with the same motion parameters. The corresponding error function is shown in figure 6.4f. The optimal motion parameters of the camera, obtained by minimizing this function, are $-\underline{U} = (1., 0., 0.)$ and $-\underline{\Omega} = (0.0000, 0.0019, 0.0001)$. These results are very close to the correct ones. Figure 6.4g shows the corresponding 'reciprocal depth' map, namely, r/Z .

7. SUMMARY

We have presented a new approach for the interpretation of optical flow fields which are induced by motion of the camera as well as motion of several rigid objects in the environment. The interpretation goals of decomposing the flow field into sets corresponding to independently moving objects, recovery of motion parameters, and estimation of relative depth of environmental surfaces were shown to be feasible. An algorithm based on our approach, was demonstrated to work with sparse, noisy and partially incorrect data, derived from both artificial and real images.

An hierarchical structure, based on four levels of organization in the flow field, has been employed. In the interpretation process units from each level are combined into larger units in the next level based on their consistency with appropriate parameter values. Thus, flow vectors, consistent with an affine transformation, are combined into one component; then, components that are compatible with the same Ψ transformation are merged into a segment; and, finally, sets of segments which satisfy the same 3-D motion parameters are hypothesized to correspond to one rigid object. The techniques for computing the parameter values in each level has been based, whenever possible, on solving linear equations derived from the least-squares criterion. Otherwise, sampling techniques combined with multi-resolution search schemes, have been employed. Combining all these techniques together, an effective and efficient algorithm has been developed.

In some situations, however, there exists an inherent ambiguity in the interpretation of noisy flow fields, as was briefly discussed and demonstrated in sections 5 and 6. In our future work we will characterize such situations and propose appropriate modifications of the interpretation goals, based on the ideas already mentioned in section 5. These modifications will be basically aimed at issues of representation of information which can be extracted even in

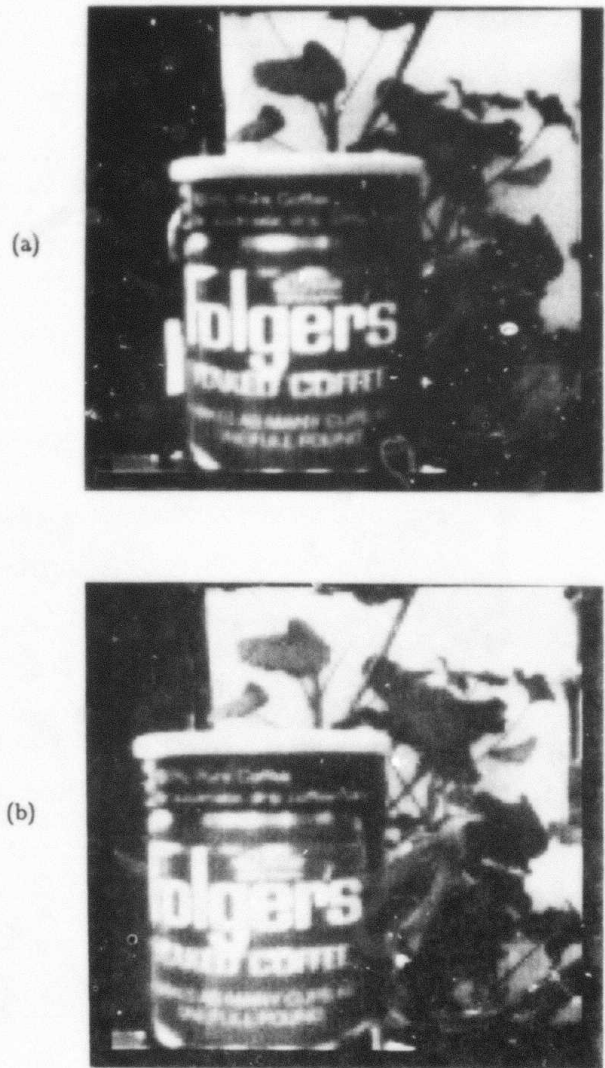
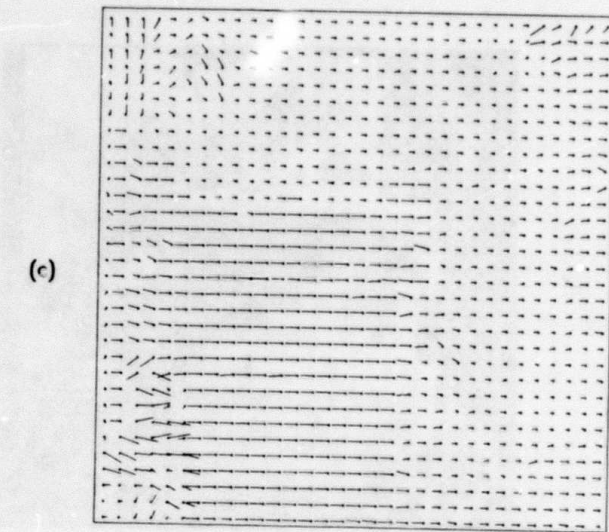
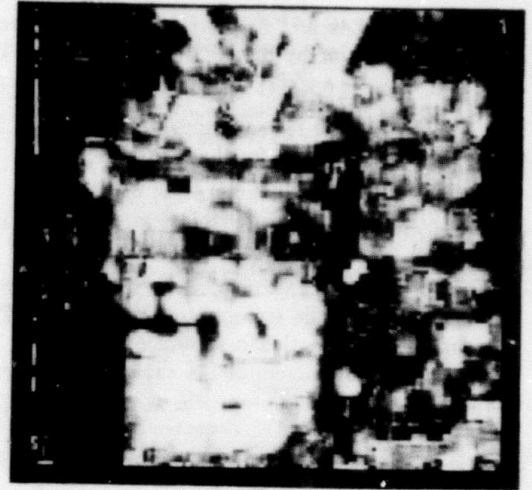


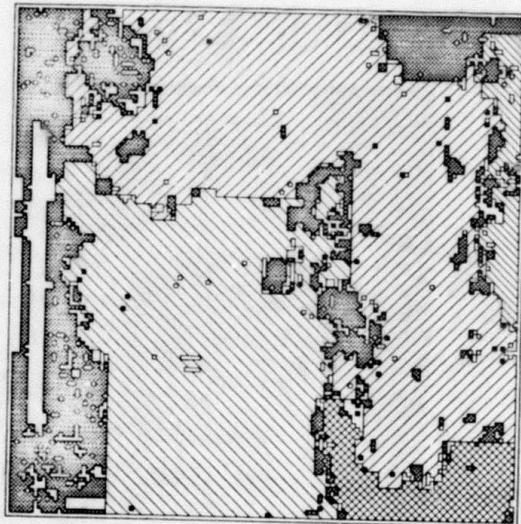
Figure 6.4: Experiment 4. (a) The first intensity image. (b) The second intensity image.



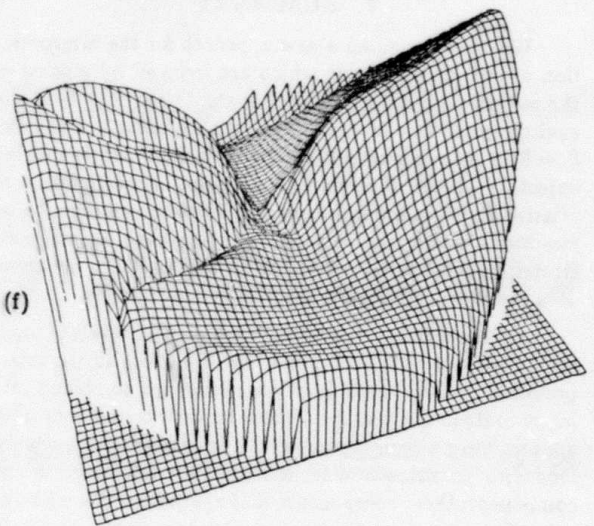
(d)



(e)



(f)



(g)

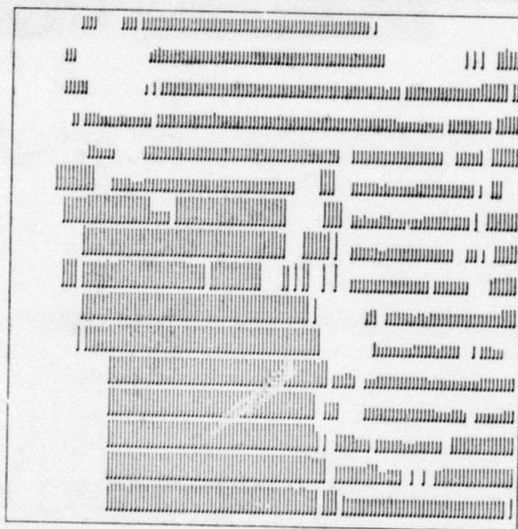


Figure 6.4, continued: (c) A 32×32 sample of the computed flow field. (d) The weight plane. High values are represented by bright gray levels. (e) Final segmentation. The white areas correspond to flow vectors assigned weight 0. The areas with the densest pattern correspond to unsegmented vectors. (f) The error function $\hat{\sigma}_1$ shown upside-down. Note the two peaks which actually correspond to the same translation, because $\hat{\sigma}_1$ is invariant to sign change in the translation vector. (g) The estimated depth function r/Z .

ambiguous cases. Integration of such information over a time sequence of flow fields may, eventually, resolve the ambiguity and result in a unique interpretation.

ACKNOWLEDGEMENTS

I am grateful to many members of the Computer Vision group at UMASS for their support and generous help. Al Hanson, Ed Riseman and P. Anandan made useful comments for improving this paper. This work was supported by DARPA under Grant N00014-82-K-0464.

REFERENCES

- [ADI83] Adiv, G., *Recovering 2-D Motion Parameters in Scenes Containing Multiple Moving Objects*, Proc. of DARPA Image Understanding Workshop, Arlington, VA (also TR 83-11, Computer and Information Science Dept., Univ. of Mass.) (1983).
- [ANA84] Anandan, P., *A Confidence Measure for Correlation Matching*, in these proceedings.
- [BAL81a] Ballard, D.H., *Parameter Networks: Towards a Theory of Low-Level Vision*, Proc. of 7th IJCAI, Vancouver, Canada (1981).
- [BAL81b] Ballard, D.H. and Kimball, O.A., *Rigid Body Motion from Depth and Optical Flow*, TR70, Computer Science Dept., Univ. of Rochester (1981).
- [BRU81] Bruss, A.R. and Horn, B.K.P., *Passive Navigation*, MIT A.I. Memo 662 (1981).
- [FAN83a] Fang, J.-Q. and Huang, T.S., *Solving Three Dimensional Small-Rotation Motion Equations*, Proc. of CVPR, Washington, D.C. (1983).
- [FAN83b] Fang, J.-Q. and Huang, T.S., *Estimating 3-D Movement of a Rigid Object: Experimental Results*, Proc. of 8th IJCAI, Karlsruhe, West Germany (1983).
- [FEN79] Fennema, C.L. and Thompson, W.B., *Velocity Determination in Scenes Containing Several Moving Objects*, CGIP 9 (1979).
- [GLA83] Glaser, F., Reynolds, G. and Anandan, P., *Scene Matching by Hierarchical Correlation*, Proc. of CVPR, Washington, D.C. (1983).
- [HAN78] Hanson, A. and Riseman, E. (Eds.), "Computer Vision Systems", Academic Press Inc., New York, NY, 1978.
- [JER83] Jerian, C. and Jain, R., *Determining Motion Parameters for Scenes with Translation and Rotation*, Proc. of the Workshop on Motion, Toronto, Canada (1983).
- [LAW82] Lawton, D.T., *Motion Analysis via Local Translational Processing*, Proc. of the Workshop in Computer Vision, Rindge, NH (1982).
- [LAW84] Lawton, D.T., *Processing Dynamic Image Sequences from a Moving Sensor*, Ph.D. Dissertation (TR 84-05), Computer and Information Science Dept., Univ. of Mass. (1984).
- [LON80] Longuet-Higgins, H.C. and Prazdny, K., *The interpretation of a Moving Retinal Image*, Proc. Roy. Soc. Lond., B 208 (1980).
- [LON81] Longuet-Higgins, H.C., *A Computer Algorithm for Reconstructing a Scene from Two Projections*, Nature 293 (1981).
- [NAG81a] Nagel, H.H., *On the Derivation of 3D Rigid Point Configurations from Image Sequences*, Proc. of PRIP, Dallas, Texas (1981).
- [NAG81b] Nagel, H.H. and Neumann, B., *On 3D Reconstruction from Two Perspective Views*, Proc. of 7th IJCAI, Vancouver, Canada (1981).
- [NEU80] Neumann, B., *Motion Analysis of Image Sequences for Object Grouping and Reconstruction*, Proc. of Pattern Recognition, Miami, Florida (1980).
- [ORO81] O'Rourke, J., *Motion Detection Using Hough Techniques*, Proc. of PRIP, Dallas, Texas (1981).
- [PRA80] Prazdny, K., *Egomotion and Relative Depth Map from Optical Flow*, Biol. Cybernetics 36 (1980).
- [PRA81] Prazdny, K., *Determining the Instantaneous Direction of Motion from Optical Flow Generated by a Curvilinearly Moving Observer*, Proc. of PRIP, Dallas, Texas (1981).
- [RIE83] Rieger, J.H. and Lawton, D.T., *Determining the Instantaneous Axis of Translation from Optic Flow Generated by Arbitrary Sensor Motion*, Proc. of the Workshop on Motion, Toronto, Canada (1983).
- [ROA80] Roach, J.W. and Aggarwal, J.K., *Determining the Movement of Objects from a Sequence of Images*, PAMI 2 (1980).
- [SLO81] Sloan, K.R., *Dynamically Quantized Pyramids*, Proc. of 7th IJCAI, Vancouver, Canada (1981).
- [TSA84] Tsai, R.Y. and Huang, T.S., *Uniqueness and Estimation of Three-Dimensional Motion Parameters of Rigid Objects with Curved Surfaces*, PAMI 6 (1984).
- [ULL79] Ullman, S., "The Interpretation of Visual Motion", MIT Press, Cambridge, Mass., 1979.
- [ULL81] Ullman, S., *Analysis of Visual Motion by Biological and Computer Systems*, Computer 14 (1981).
- [WAX83] Waxman, A.M. and Ullman, S., *Surface Structure and 3-D Motion from Image Flow: A Kinematic Approach*, CAR-TR-24, Center for Automation Research, Univ. of Maryland (1983).
- [WIL81] Williams, T.D., *Computer Interpretation of a Dynamic Image from a Moving Vehicle*, Ph.D. Dissertation (TR 81-22), Computer and Information Science Dept., Univ. of Mass. (1981).

DYNAMIC STEREO: PASSIVE RANGING TO MOVING OBJECTS FROM RELATIVE IMAGE FLOWS

Allen M. Waxman and Sarvajit S. Sinha

Computer Vision Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742

ABSTRACT

A new concept in passive ranging to moving objects is described which is based on the comparison of multiple image flows. It is well known that if a static scene is viewed by an observer undergoing a known relative translation through space, then the distance to objects in the scene can be easily obtained from the measured image velocities associated with features on the objects (i.e. motion stereo). But in general, individual objects are translating and rotating at unknown rates with respect to a moving observer whose own motion may not be accurately monitored. The net effect is a complicated image flow field in which absolute range information is lost. However, if a second image flow field is produced by a camera whose motion through space differs from that of the first camera by a known amount, the range information can be recovered by subtracting the first image flow from the second. This "difference flow" must then be corrected for the known relative rotation between the two cameras, resulting in a divergent relative flow from a known focus of expansion. This passive ranging process may be termed *Dynamic Stereo*, the known difference in camera motions playing the role of the stereo baseline. We present the basic theory of this ranging process, along with some examples for simulated scenes. Potential applications are in autonomous vehicle navigation (with one fixed and one movable camera mounted on the vehicle), coordinated motions between two vehicles (each carrying one fixed camera) for passive ranging to moving targets, and in industrial robotics (with two cameras mounted on different parts of a robot arm) for intercepting moving workpieces.

1. INTRODUCTION

Passive ranging by triangulation methods, employed so successfully by humans, has received much attention in the machine vision literature in recent years [1]. It is obvious that the ability to recover absolute range to objects in a scene would be important in a variety of robotic applications. And passive techniques to do so are

certainly of interest. To date, only two basic methods of passive ranging have been discussed, "static stereo" (employing two cameras separated by a known baseline) and "motion stereo" (utilizing a single camera moving in a known way through a stationary scene). In this paper we introduce a new concept in passive ranging to moving objects, termed "dynamic stereo," which is based on the comparison of multiple image flows.

By far, most of the literature on passive ranging has been concerned with the difficult "correspondence problem" associated with the assignment of stereo disparities (see the many references cited in [1]). In addition to the traditional method of intensity correlation between images, much interest has been paid to the theory of Marr and Poggio [2], with its implementation by Grimson [3] and recent insights of Nishihara [4], as well as the theory of Mayhew and Frisby [5]. The use of more than two camera locations, to aid in solving the correspondence between images, has been approached in different ways by Moravec [6] and Tsai [7]. Nevertheless, solution of this correspondence problem remains a computationally expensive and slow process. Moreover, a maximum ranging distance is implied by the finite resolution of the cameras and the statically configured baseline between cameras.

In principle, the difficulties encountered with "static stereo" can be overcome using "motion stereo." Here, a single camera is moved through space in a known way, while imaging a stationary scene. The result is that, over a period of time, the camera traverses a known physical baseline of arbitrary length. In addition, correspondence is established by tracking features over the small inter-frame distances to build up effective stereo disparities, which then yield range values [8, 9]. In practice, problems can arise from inaccuracies in the camera motion parameters. However, a more fundamental limitation is the necessity of a stationary scene. If, while the camera is moving from one end of the baseline to the other, objects in the scene are also moving but with unknown velocities, then the relative motions between camera and objects would not be known and hence, the physical baseline would be undetermined. In such cases, absolute range information is lost, though relative range (such as object surface slopes) and scaled motion parameters can

Support of the Defence Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAK70-83-K-0018 (DARPA Order 3206) is gratefully acknowledged.

Dynamic Stereo can be viewed as an extension of motion stereo, applicable to scenes containing moving objects. It employs two cameras in *known relative motion*, both imaging a scene containing independently moving objects (which are assumed rigid). The relative rigid body motions between objects and cameras generate image flows (of feature points and contours) at each camera. Differences between the two flow fields are mainly due to the known relative motion between the two cameras. This fact will be exploited below in order to recover absolute range to the objects in an evolving scene. Dynamic stereo can then be used in conjunction with the image flow derived from a single camera in order to recover surface shape as well as absolute motion parameters for objects in the scene [11].

This paper presents the basic theory of Dynamic Stereo along with several simulated examples. The concepts associated with "relative image flows" are described in Section 2, which follows. Section 3 then addresses the recovery of range to moving objects using these relative flows. Filtering techniques to reduce the effects of noise on the required image velocities are discussed as well. Concluding remarks are presented in Section 4.

The flow fields measured at each camera correspond to the time-varying projection of object surface texture, due to the relative rigid body motions between objects in the scene and the cameras. The equations relating image velocity to relative space motion and distance to points in the scene have been derived in other studies (e.g. [10 - 12]); they are given by

Figure 1 illustrates the coordinate systems of the observer (X, Y, Z), to whom the relative translations (V_X, V_Y, V_Z) and rotations ($\Omega_X, \Omega_Y, \Omega_Z$) are ascribed (and which may differ for each rigid object in the scene), and his image plane (x, y) which has been reinverted and scaled to a focal length of unity. As the directions to points (or objects) in the scene are specified by their image coordinates (x, y), their absolute range is determined by the component of distance Z , along the observer's line of sight. It is seen from equations (1) that the distance Z appears in ratio with the translational motion parameters. Clearly, if the motion parameters were all known (e.g. a camera moving through a stationary scene), the distance Z could be obtained directly from measured image velocities; this is simply "motion stereo." But if the objects are also moving, then these relative motion parameters are unknown as is the distance Z to points (or "scale factor" Z_0 , slopes and curvatures in the case of surface patches). The theory of single image flows addresses this problem of recovering both surface structure and space motion [10 - 13]; however, solutions are obtained in a form which are scaled by the factor Z_0 . That is, absolute range is not recoverable from single image flows.

$$v(x, y) = \frac{1}{Z(x, y)} T(x, y) \cdot V + R(x, y) \cdot \Omega, \quad (2)$$

where the elements of the translation and rotation matrices, \vec{T} and \vec{R} , are functions only of the image coordinates, and may be read directly from equations (1).

In a dynamic stereo configuration, each camera has its own set of relative motion parameters (for each object in the scene), and generally, the two image planes are not coincident. However, we can configure the two cameras such that they come very close to each other at some time during their own relative motion. At that instant, the two image planes are nearly coincident and correspondence should be easily established. If we refer the measured image velocities to that moment in time, then we may treat the matrices $\vec{T}(x, y)$ and $\vec{R}(x, y)$ for the two cameras as identical, as if the two image planes were indeed coincident momentarily. (More will be said of the residual effects of a small but finite baseline in Section 3.4 below.) Now if the relative motion between cameras is given by $\Delta V \equiv (\Delta V_X, \Delta V_Y, \Delta V_Z)$ and $\Delta \Omega \equiv (\Delta \Omega_X, \Delta \Omega_Y, \Delta \Omega_Z)$, then the second camera sees an augmented image flow of $v + \Delta v$, where from (2), the "difference flow" Δv is given by

$$\Delta v(x, y) = \frac{1}{Z(x, y)} \vec{T}(x, y) \cdot \Delta V + \vec{R}(x, y) \cdot \Delta \Omega. \quad (3)$$

As the distance information is contained in the translational term in (3), we can bring the known rotational term to the left-hand side and define a "relative flow" or "modified difference flow" as

$$\Delta w \equiv \Delta v - \vec{R} \cdot \Delta \Omega = \frac{1}{Z} \vec{T} \cdot \Delta V. \quad (4)$$

This relative flow takes the form of a divergent flow field with a known focus of expansion (as long as $\Delta V_Z \neq 0$), as is seen from writing the individual components of (4),

$$\Delta w_x = \frac{\Delta V_Z}{Z} (x - x_{foe}) \quad (5a)$$

$$\Delta w_y = \frac{\Delta V_Z}{Z} (y - y_{foe}), \quad (5b)$$

where

$$x_{foe} \equiv \frac{\Delta V_X}{\Delta V_Z} \quad \text{and} \quad y_{foe} \equiv \frac{\Delta V_Y}{\Delta V_Z}. \quad (5c,d)$$

Figures 2a, b, c illustrate examples of such divergent relative flows. In each case, a simulated scene is constructed and feature points are marked. The two sets of space motion parameters are selected, corresponding to the relative motion between objects and each of the two cameras (the difference between these sets of parameters being the relative motion between cameras). The individual flows of the feature points are displayed along with the divergent relative flow. No noise effects were introduced into this simulation. Figures 2 were created with the *Image Flow Simulator* [14].

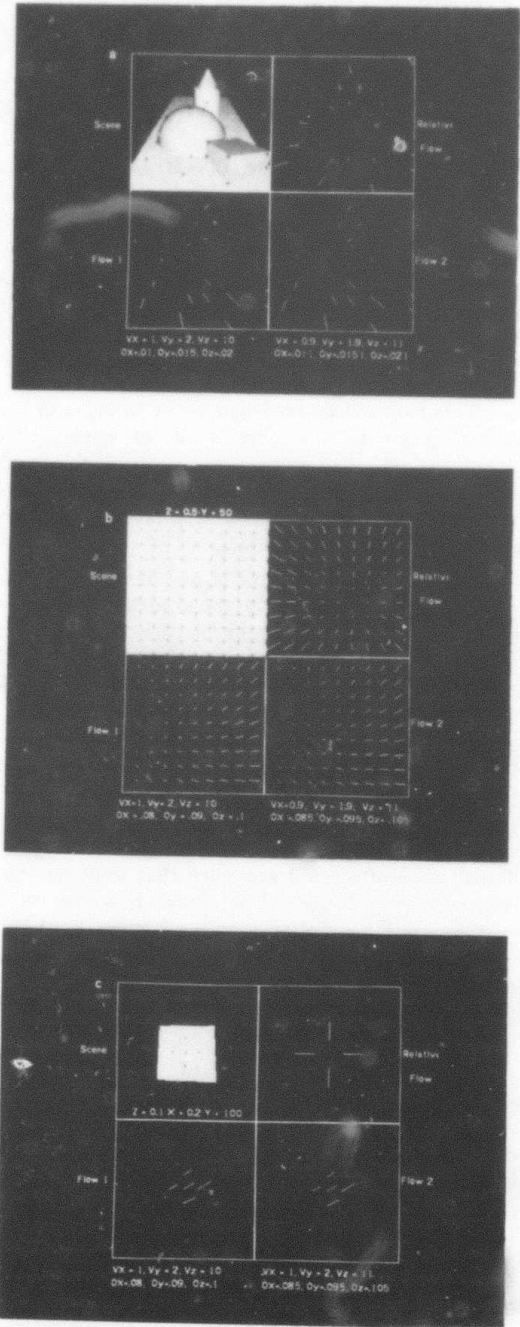


Fig. 2 - Examples of the relative flow fields for:
a) a complex scene with feature points marked,
b) a planar surface with a grid of feature points,
c) a small planar patch with few feature points.
For each case the simulated scene is shown, along with the ideal flows for each set of camera motion parameters and the divergent relative flow.

3. RANGING TO MOVING OBJECTS

Equations (4) and (5) form the basis of the method for recovering range to points (on moving objects in the scene) and planes (surface patches on moving objects). If it were not for the effects of noise on measured image velocities, the method would be simple and straightforward, as will be presented in Section 3.1 below. The inevitable effects of digitization error and noise have led us to explore two filtering methods; "radial flow filtering" as motivated by the divergent nature of the relative flow noted in (5), and "second-order flow filtering" which stems from the *Velocity Functional Method* developed by Waxman and Wahn [12]. These techniques are described, along with examples, in Sections 3.2 and 3.3, respectively. Section 3.4 considers the effects of the finite baseline between cameras, at their closest approach.

3.1. Ranging to Points and Planes

Given the measured image velocities of corresponding features on both image planes (determined when the two cameras are at closest approach), we form the measured difference flow values Δv . Then, according to the definition in (4), we can compute the relative flows by correcting these difference values for the known relative rotation between cameras,

$$\Delta w_x = \Delta v_x - [xy \Delta \Omega_X - (1 + x^2) \Delta \Omega_Y + y \Delta \Omega_Z], \quad (6a)$$

$$\Delta w_y = \Delta v_y - [(1 + y^2) \Delta \Omega_X - xy \Delta \Omega_Y - x \Delta \Omega_Z]. \quad (6b)$$

According to equations (5a-d), the ideal relative flow diverges from (or converges to) a known focus located at (x_{foc}, y_{foc}) . Thus, we can define the "radial relative flow" as simply

$$\Delta w_r \equiv \left\{ (\Delta w_x)^2 + (\Delta w_y)^2 \right\}^{1/2}. \quad (7a)$$

Then according to (5), we can solve for the range Z ,

$$Z(x, y) = \frac{\Delta V_Z}{\Delta w_r(x, y)} \left\{ (x - x_{foc})^2 + (y - y_{foc})^2 \right\}^{1/2}. \quad (7b)$$

This result applies to individual feature points in the scene whose relative flow has been derived from measured image velocities on both cameras. All terms on the right-hand side of (7b) are either known or measured.

If a number of feature points are believed to be the images of points on a planar surface in space, then their individual range values $Z(x, y)$ can be used to fit a planar surface. Alternatively, the parameters of the surface can be obtained collectively from (7b). A planar surface in space, $Z = Z_0 + pX + qY$, can be written exactly in image coordinates as $Z = Z_0(1 - px - qy)^{-1}$. Inverting (7b) then yields

$$\frac{1}{Z_0} (1 - px - qy) = \frac{\Delta w_r(x, y)}{\Delta V_Z} \left\{ (x - x_{foc})^2 + (y - y_{foc})^2 \right\}^{-1/2}. \quad (7c)$$

This equation can serve as the basis for a linear least-squares approach to determine the parameters of the plane, Z_0^{-1} , p/Z_0 and q/Z_0 .

3.2. Radial Flow Filtering

In the absence of noise and digitization effects, equations (7) are exact and yield perfect results, putting aside for the moment the issue of finite separation between cameras at all times. In our simulation, we are able to explore the effects of noise by perturbing the individual flow values before forming the difference flow. We considered a uniform distribution of noise, up to a specified percentage, superposed on the individual components of image velocity.

The effects of noise on estimated image velocities v are amplified by the differencing procedure in obtaining Δv . Since the order of magnitude of v is $|V|/Z$, while that of Δv is $|\Delta V|/Z$, the noise effects are amplified by the ratio $|V|/|\Delta V|$ when referred to Δv . Clearly, it is important to have as large a relative velocity between cameras ΔV as is possible. This translates over time to building up as large a separation between cameras as is practical.

One simple thing that can be done to reduce the effects of noise is to perform a "radial filtering" on the relative flow. This notion stems from the divergent nature of the ideal relative flow discussed in Section 2. As definition (7a) is meant to imply that the ideal relative flow should consist of vectors emanating radially from a known focus of expansion, we can impose this constraint on the relative flow derived from noisy velocity measurements. That is, we consider only that component of the relative flow which points radially from a known focus. The orthogonal component (or "azimuthal relative flow") can be used to ascertain the magnitude of the noise; it vanishes in the limit of ideal velocity measurements.

Figure 3 illustrates the simulated effects of noise (10%) on the image velocities and the relative flow which results, in the case of a planar surface. By using only the radial component of the relative flow, one essentially reduces the noise by a factor of two. As one might expect, the errors in range should scale like $(\% \text{ noise}) \times |V|/|\Delta V|$. The results of our simulations bear this out. For a ratio $|\Delta V|/|V|$ of about 1/10, reasonable range estimates could be found only when the noise imposed was less than a few percent. In the case of planar surfaces, the same is true for recovery of the scale factor Z_0 . The slopes of the surface, which describe the differential changes in range, are extremely sensitive to noise, as might be expected. Their determination requires noise below one percent. In practice, one

could not expect surface slope recovery from dynamic stereo; the individual image flows seem more appropriate for this task [10, 12]. However, qualitative recovery of range from relative flows does appear feasible. For the case of ranging to surfaces, one can do better still, by using the following filtering technique.

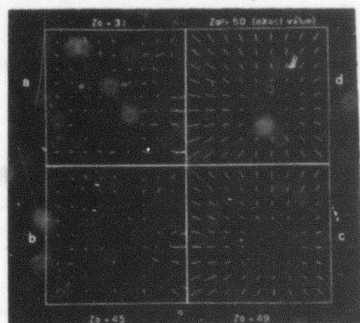


Fig. 3 - Effects of 10% noise on feature point velocities for the planar surface example (value found for Z_0 is indicated in each case):

- a) the relative flow obtained directly from the noisy velocities,
- b) the relative flow after "radial flow filtering",
- c) the relative flow obtained from "second-order flow filtering",
- d) the ideal relative flow.

3.3. Second-Order Flow Filtering

The drawback of "radial flow filtering" is that it operates on the relative flow rather than on the individual flows preceding the differencing operation. As this differencing procedure tends to amplify noise, the radial filtering is not sufficient to recover entirely from the process. What is needed is a filtering process which reduces the noise effects on the individual flows, before their difference is taken. When the image velocities utilized arise from features on objects at various depths moving with different space motions, no smoothing of the individual flows can be performed as the image velocities themselves are unrelated. However, when features (points or contours) arising from single objects are isolated, their image velocities constitute a locally second-order flow field [11, 12, 13], and this can be used to filter out the effects of noise on the individual image flows, before differencing.

The representation of image flows in terms of second-order flow fields has been termed the *Velocity Functional Method* by Waxman and Wahn [12]. It is a globally valid representation in the case of planar surfaces [12], and a locally valid one for curved surfaces [13]. The coefficients of the second-order polynomials are simply related to the derivatives of the flows via their truncated Taylor series about a local origin. Let $v_k(x, y)$ be the flow in image k ($k = 1, 2$), and $v_k^{(i,j)}$ be its i^{th} partial derivative with respect to x and j^{th} partial with respect to y (where $i + j \leq 2$), evaluated at a local origin in an image neighborhood. We have, locally

$$v_k = \sum_{i=0}^2 \sum_{j=0}^2 v_k^{(i,j)} \frac{x^i}{i!} \frac{y^j}{j!} \quad (8)$$

The coefficients are obtained from a linear least-squares fit to measured velocities in an image neighborhood. For point features, both components of image velocity are available, and (8) then provides two constraints. In the case of contours evolving in geometry over time, only a normal velocity (perpendicular to the contour) is perceptible, and (8) provides only one constraint in the form of a linear combination of the two velocity components (cf. [12] for details). When only evolving contours are used, a minimum structure is required of a single contour if it is to yield all twelve coefficients in (8). It is generally more robust to use several contours in a given neighborhood [12]. The flow field, as reconstructed from (8), is typically much cleaner than the measured image velocities themselves, and is therefore more suitable for forming the difference flow.

With expressions in the form of (8) for corresponding neighborhoods in the two images, we can form the difference flow simply by subtracting the coefficients of the respective representations. Then, correcting for the relative rotations as in (6), we have for the components of relative flow

$$\Delta w_x = \sum_{i=0}^2 \sum_{j=0}^2 [v_{x,2}^{(i,j)} - v_{x,1}^{(i,j)}] \frac{x^i}{i!} \frac{y^j}{j!} - [xy \Delta \Omega_X - (1 + x^2) \Delta \Omega_Y + y \Delta \Omega_Z], \quad (9a)$$

$$\Delta w_y = \sum_{i=0}^2 \sum_{j=0}^2 [v_{y,2}^{(i,j)} - v_{y,1}^{(i,j)}] \frac{x^i}{i!} \frac{y^j}{j!} - [(1 + y^2) \Delta \Omega_X - xy \Delta \Omega_Y - x \Delta \Omega_Z]. \quad (9b)$$

Equations (9) for the relative flow are, themselves, second-order polynomials in the image coordinates. They can be equated term-for-term with the expanded form of equations (5), written for a surface $Z(x, y)$. In the case of a planar surface, equations (5) yield

$$\Delta w_x = \frac{1}{Z_0} \left\{ -\Delta V_X + (p \Delta V_X + \Delta V_Z) x - (q \Delta V_X) y - (p \Delta V_Z) x^2 - (q \Delta V_Z) xy \right\}, \quad (10a)$$

$$\Delta w_y = \frac{1}{Z_0} \left\{ -\Delta V_Y + (p \Delta V_Y) x + (q \Delta V_Y + \Delta V_Z) y - (q \Delta V_Z) y^2 - (p \Delta V_Z) xy \right\}. \quad (10b)$$

By comparing the coefficients of (10) with those determined in (9), one sees that the parameters of the plane (Z_0, p, q) can be determined from the zeroth-order and first-order terms directly. Non-planar surfaces will lead to modifications of the second-order terms (and generate higher-order terms as well). As one would expect the zeroth-order terms of (9) to be the most precise, it should be clear from (10) that Z_0 can be recovered most accurately if ΔV_X and ΔV_Y are non-zero. This is borne out by our simulations in which reasonable results for Z_0 could be obtained even at 10% noise levels, with $|\Delta V|/|V| \approx 0.1$.

In our simulations, once the coefficients in (9) have been determined, we have found it more convenient to return to (7) and solve for the parameters of the plane by a least-squares procedure (having sampled (9) over a sparse grid), rather than literally identify coefficients with equations (10). In general, it is best to have all components of ΔV non-zero. By filtering the individual flows before differencing, one is able to recover surface scale factors Z_0 even in the presence of noisy image velocities. Figure 3 also shows the relative flow for a plane after second-order filtering; it is clearly a divergent flow. Figure 4 illustrates the case of two elliptic contours on a planar surface, with their individual normal flows, one of the full flows recovered by the *Velocity Functional Method*, and the relative flow from which Z_0 is recovered. This same example, with normal velocities perturbed by 50%, is shown in Figure 5.

In our simulations we have tried to ascertain the effects of noise and field of view on recovery of Z_0 . In general, when utilizing second-order flow filtering, we have found recovery of Z_0 from *point feature velocities* to be possible at 10% noise levels, while utilizing *evolving contours* allows recovery of Z_0 even with 50% perturbations to the normal flow around the contour. Moreover, ranging is possible down to fields of view of about 3° , below which variations in image velocity become so small that the method becomes unstable at noise levels exceeding about 5%. Typically, surface slopes cannot be reliably determined from dynamic stereo in the presence of noise; however, they can be found from the analysis of single image flows [10, 12] (and then utilized with dynamic stereo to recover Z_0 somewhat more accurately). Finally, we attempted to recover Z_0 to curved surfaces, treating them as if they were planar. As one might expect, when variation in range over a surface is small compared to absolute range to that surface, the method succeeds in recovering the scale factor Z_0 . The slopes obtained can be used to find the approximate distance to individual feature points on the curved surface. However, when range variations are large (as compared to a planar surface at comparable Z_0), then the recovery of Z_0 can be rather sensitive to noise. This result would favor small fields of view, where substantial range variation is unlikely; although too small a field of view (below about 3°) leads to noise sensitivity as well.

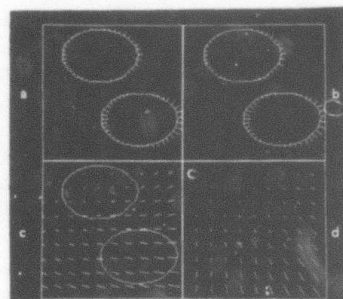


Fig. 4 - The use of "normal flow" along contours on a planar surface $Z = Z_0 + 0.1X + 0.2Y$:
a & b) the ideal normal flows for each set of camera motion parameters,
c) the full flow for (a) recovered by the Velocity Functional Method,
d) the relative flow found from the recovered second-order flows.

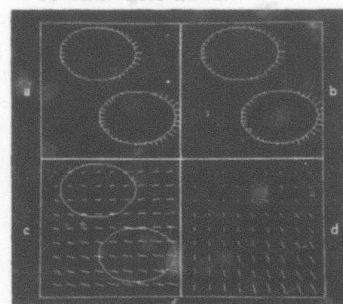


Fig. 5 - Same as Fig. 4, but with 50% noise superposed on the normal flows.

3.4. Effects of the Finite Baseline

The underlying basis of Dynamic Stereo is the comparison of image flows obtained from two cameras in known relative motion. This notion is implicit in the "relative flow" discussed in Section 2. But in forming the required "difference flow", we treated the translation and rotation matrices of the two cameras as identical, which is equivalent to treating the two image planes as coincident at the time of closest approach between cameras (see the comments preceding equation (3)). Since the two cameras must be physically separated by some small but finite baseline, even at their closest approach, it is really only a simplification to treat the two image planes as momentarily registered. That is, the image coordinates of a given feature are not exactly the same on the two image planes. The remaining disparity between coordinates is a function of the distance to the feature, exactly like the case of "static stereo". However, this disparity should not be a real problem since keeping this baseline small will lead to an insignificant disparity, except for objects which are extremely close.

In theory, we can account for the finite baseline at closest approach by incorporating a small correction to the image coordinates of features imaged by the second

camera. This correction, being the range dependent disparity, can be treated as a small perturbation to the ranging formulas derived above. This would lead to a "successive approximation" scheme for recovering Z_0 , in which the theory developed here would serve as the lowest-order approximation, to which corrections can be applied. However, given the inaccuracies associated with noisy flow values, such an iteration scheme hardly seems warranted.

We have investigated the effects of a finite baseline at closest approach with our Image Flow Simulator [14]. An artificial scene is constructed and motion parameters are assigned to an observer, yielding the first image flow field (obtained from second-order flow filtering of point feature velocities). The second flow field is obtained using the same scene shifted by a small amount in a direction parallel to the image plane (in order to simulate camera separation), and different motion parameters are assigned to the observer. The two flow fields are then differenced and corrected for relative rotation in order to recover range. In the case of a planar surface patch, in the absence of noise, a baseline to range ratio of 1/1000 led to a 2% range error. A ratio of 5/1000 yields about a 10% error, which is comparable to the accuracy obtainable with 10% noise. It would seem that this ratio is reasonable when ranging to objects at several hundred feet or more.

4. CONCLUDING REMARKS

We have introduced a new concept in passive ranging, termed *Dynamic Stereo*, which is based on the comparison of image flow fields obtained from two cameras in known relative motion. The method is designed for ranging to moving objects in an evolving scene. For stationary scenes, this technique reduces to conventional "motion stereo". In this paper we have developed the basic theory and studied the effects of noisy image velocities and field of view, with the aid of an Image Flow Simulator [14]. These simulations suggest that ranging to points, planes and curved surfaces is feasible, even in the presence of 10% noise. Two methods for filtering the noise were also introduced, "Radial Flow Filtering" of the corrected difference flow and "Second-Order Flow Filtering" prior to differencing. The second method is preferred, but may only be applied to image velocities generated by a single surface patch.

It is anticipated that Dynamic Stereo may have a number of interesting applications. First, it may be used in conjunction with single image flow analysis, providing the scale factor required for the complete recovery of object structure and space motion from time-varying imagery [11]. Second, it may prove useful in industrial robotics for handling moving workpieces in an evolving workplace. Two cameras would have to be configured on different parts of the robot arm so that they will experience a relative motion. Third, Dynamic Stereo has potential application to autonomous vehicle navigation for

ranging to other moving objects in the scene. Possible configurations are two cameras in relative motion on a land vehicle, or one camera on each of two aircraft in known relative motion.

To appreciate the distance scales involved with this approach to passive ranging, we can try to scale up our simulation examples to the case of land vehicles. A vehicle traveling at 50 km/hr moves about 50 feet in one second. Upon the vehicle there is mounted a fixed camera and a sliding camera which moves about 5 feet during the one second interval. The scale factors recovered would then correspond to a distance of the order of 500 feet.

Our simulations indicate that best results are obtained when the sliding camera moves at an angle with respect to the direction of vehicle motion. Implementation of this new ranging technique on actual evolving scenes, utilizing cameras in relative motion, is planned for the near future.

REFERENCES

- [1] R.A. Jarvis, "A Perspective on Range Finding Techniques for Computer Vision", *IEEE Trans. Pattern Analysis and Machine Intelligence* **5**, pp. 122 - 139, 1983.
- [2] D. Marr & T. Poggio, "A Computational Theory of Human Stereo Vision", *Proc. Roy. Soc. London* **B204**, pp. 301 - 328, 1979.
- [3] W.E.L. Grimson, *From Images to Surfaces* (Cambridge: MIT Press), 1981.
- [4] H.K. Nishihara, "PRISM: A Practical Real-Time Imaging Stereo Matcher", *MIT Artificial Intelligence Memo 780*, 1984.
- [5] J.E.W. Mayhew & J.P. Frisby, "Psychophysical and Computational Studies Towards a Theory of Human Stereopsis", *Artificial Intelligence* **16**, pp. 349 - 385, 1981.
- [6] H.P. Moravec, *Robot Rover Visual Navigation* (Ann Arbor: UMI Research Press), 1981.
- [7] R.Y. Tsai, "Multiframe Image Point Matching and 3-D Surface Reconstruction", *IEEE Trans. Pattern Analysis and Machine Intelligence* **5**, pp. 159 - 173, 1983.
- [8] R. Nevatia, "Depth Measurement from Motion Stereo", *Computer Vision, Graphics & Image Processing* **9**, pp. 203 - 214, 1976.
- [9] T.D. Williams, "Depth from Camera Motion in a Real World Scene", *IEEE Trans. Pattern Analysis and Machine Intelligence* **2**, pp. 511 - 516, 1980.
- [10] A.M. Waxman & S. Ullman, "Surface Structure and 3-D Motion from Image Flow: A Kinematic Analysis", *Univ. of Maryland, Center for Automation Research Tech. Report 24*, October 1983.
- [11] A.M. Waxman, "An Image Flow Paradigm", *Univ. of Maryland, Center for Automation Research Tech. Report 45*, February 1984, and *Proc. 2nd IEEE Workshop on Computer Vision: Representation and Control*, pp. 49 - 57, April 1984.
- [12] A.M. Waxman & K. Wahn, "Contour Evolution, Neighborhood Deformation and Global Image Flow: Planar Surfaces in Motion", *Univ. of Maryland, Center for Automation Research Tech. Report 58*, April 1984.
- [13] K. Wahn & A.M. Waxman, "Contour Evolution, Neighborhood Deformation and Local Image Flow: Curved Surfaces in Motion", *Univ. of Maryland, Center for Automation Research Tech. Report (in preparation)*.
- [14] S. Sinha & A.M. Waxman, "An Image Flow Simulator", *Univ. of Maryland, Center for Automation Research Tech. Report 71*, July 1984.

The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images

Martin Herman
Takeo Kanade

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

The 3D Mosaic system is a vision system that incrementally reconstructs complex 3D scenes from multiple images. The system encompasses several levels of the vision process, starting with images and ending with symbolic scene descriptions. This paper describes the various components of the system, including stereo analysis, monocular analysis, and constructing and modifying the scene model. In addition, the representation of the scene model is described. This model is intended for tasks such as matching, display generation, planning paths through the scene, and making other decisions about the scene environment. Examples showing how the system is used to interpret complex aerial photographs of urban scenes are presented.

Each view of the scene, which may be either a single image or a stereo pair, undergoes analysis which results in a 3D wire-frame description that represents portions of edges and vertices of objects. The model is a surface-based description constructed from the wire frames. With each successive view, the model is incrementally updated and gradually becomes more accurate and complete. Task-specific knowledge, involving block-shaped objects in an urban scene, is used to extract the wire frames and construct and update the model.

1. Introduction

It is important for a general vision system to derive three-dimensional (3D) information about a given scene from images and store the information in a coherent manner so that it can be used for various matching, planning, and display tasks. Our goal in developing the 3D Mosaic system has been to build a full vision system, that is, one that goes all the way from images to symbolic 3D descriptions. Further, we wanted to investigate this process in the context of complex scenes. The result is really a first pass at such a system, and provides us with a better understanding of the components required. This paper describes the system and presents examples of how it is used to interpret complex aerial photographs of urban scenes.

2. The 3D MOSAIC System

The goal of the 3D Mosaic system is to obtain an understanding of the 3D configuration of surfaces and objects in a scene. The significance of this goal may be demonstrated by the following tasks.

1. **Model-based image interpretation.** A known 3D scene model can provide significant aid in interpreting arbitrary images of the scene [7, 19, 24]. The 3D Mosaic system performs the task of acquiring such a model of the scene.
2. **3D change detection.** Change detection is a task that determines how the geometry and structure of a scene

changes over time. The conventional approach to this task involves comparing and detecting changes in images. However, because of different viewpoints and lighting conditions, changes in the images do not necessarily correspond to changes in the geometry and structure of the scene. If 3D scene descriptions were obtained from the images first, such descriptions could be compared in 3D to determine changes in the scene.

3. **Simulating the appearance of the scene.** If a 3D description of the scene were to be obtained, displays as seen from arbitrary viewpoints could be generated from it. This is useful for tasks such as familiarizing personnel with a given area, and flight planning by generating the scene appearance along hypothetical flight paths.
4. **Robot navigation.** Three-dimensional descriptions of complex environments may be used to make decisions dealing with path planning or determining which parts of the environment to analyze in more detail.

The 3D Mosaic system deals with complex, real-world scenes (e.g., Fig. 4). That is, the scenes contain many objects with a variety of shapes, the object surfaces have a variety of textures and reflectance characteristics, and the scenes are imaged under outdoor lighting conditions. Because of the complexity, there are many difficulties in interpreting the images, including:

1. Any particular image contains only partial information about the scene because many surfaces are occluded.
2. Even portions of the scene that are visible are often difficult to recover. For example, surfaces with dark shadows cast across them, or with highlights, may be difficult to interpret. Highly oblique surfaces may be difficult to analyze if their resolution in the image is poor.

Our approach to the problems of complexity is to use multiple images obtained from multiple viewpoints. This approach aids interpretation in two ways. First, surfaces occluded in one image may become visible in another. Second, features of surfaces that are difficult to analyze and interpret in one image (such as scene edges and texture) may become more apparent in another image because of different viewpoint and/or lighting conditions.

2.1. Incremental Approach

A large number of views will, in general, be required to obtain a fully accurate and complete description of a complex scene. Typically, all these views will not be simultaneously available, while some may never

become available. Many of them will only be obtained gradually through interaction with the scene environment. Our system must therefore have the ability to utilize partial descriptions and incrementally update them with new information whenever a new view happens to become available. As a practical example, consider a robot (perhaps a mobile ground robot or an automatically guided airplane) which is attempting to navigate through an unknown environment. The robot would sequentially acquire images of the environment as it moves about. Information derived from each new image would serve to update its internal model, and this partial model would be used to decide where to go next, or where to analyze in more detail.

We have adopted an approach in which the 3D scene model is incrementally acquired over the multiple views. The views of the scene are sequentially acquired and processed. Partial 3D information is derived from each view. The initial model is constructed from 3D information obtained from the first view, and represents an initial approximation of the scene. As each successive view is processed, the model is incrementally updated and gradually becomes more accurate and complete.

Most previous research efforts at acquiring 3D scene descriptions from multiple views have dealt with relatively simple scenes in controlled environments [2, 8, 9, 18, 22, 25]. This has led, in some cases, to only utilizing occluding contours in the image to form the 3D description [2, 8, 9, 18]. The work of Moravec [20] deals with complex indoor and outdoor scenes, but the 3D descriptions generated by his system consist of sparse sets of feature points. Our system, on the other hand, generates full, surface-based descriptions.

2.2. Overview

A flowchart for the 3D Mosaic system, showing the major modules and data structures, is displayed in Fig. 1. The input is a new view of the scene, which may be either a stereo image pair or a single image. The stereo pair undergoes stereo analysis, while the single image undergoes monocular analysis. The purpose of these analyses is to obtain 3D scene features such as portions of surfaces, edges, and corners.

The central scene model is a surface-based description which is constructed and modified from these features. Before modifications to the scene model can occur, the 3D features from the new view must be matched to the current model. The scene model may, at any point along its development, be used for tasks such as image interpretation, planning, or display generation. A new view may then be acquired which may further modify the model.

For example, when the stereo analysis component is applied to the images in Fig. 4, the result is the set of wire frames in Fig. 9. The scene model constructed from these wire frames is shown in Fig. 20. When the monocular analysis component is applied to the image in Fig. 10, the result is the set of wire frames in Fig. 17. These, in turn, are converted into the scene model in Fig. 21. Finally, the result of modifying the model in Fig. 20 with a new view is shown in Fig. 27.

3. Stereo Analysis

Most stereo matching methods involve matching low-level image features, such as image intensities [3, 13, 17, 21] or image edge points [3, 12, 21]. Points to be matched may also be chosen as "interesting points", e.g., those with high variance in all directions [6, 20]. Our method involves matching structural features -- i.e., junctions -- extracted from the images. There are several reasons for this.

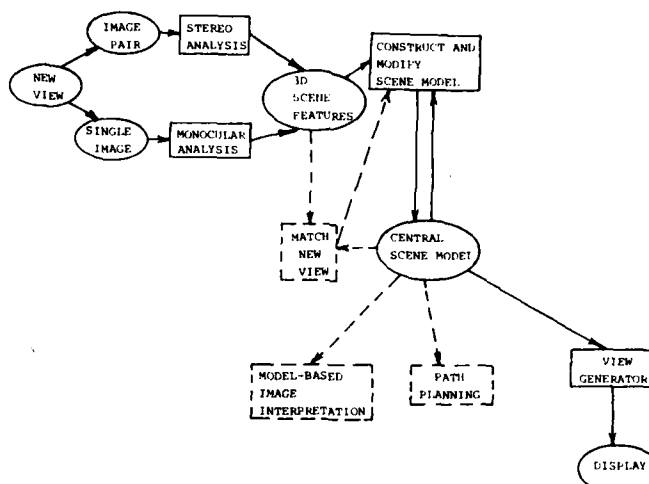


Figure 1: 3D Mosaic flowchart. The dashed lines represent components that have not yet been implemented; the solid lines represent components already implemented.

First, feature-based matching results in more accurate 3D positions for occlusion boundaries than gray scale area matching. Second, by extracting 3D information dealing with scene vertices and edges emanating from them, we obtain portions of boundaries of scene buildings, particularly building corners. These boundaries are then used to construct 3D approximations of the buildings.

Finally, because of our wide-angle stereo images, there are large disparity jumps and large portions of the scene are visible in one image but not the other. Because most stereo systems do not distinguish these from other regions of the image, they try to find matches for them and therefore have trouble [3, 5, 6, 12, 13, 17].

In our approach, rather than attempting to find matches for scene faces occluded in one of the images, we match face boundaries visible in both images. We do this by explicitly taking into account the way junction appearances change from one image to the other, using the knowledge that in urban scenes, roofs of buildings tend to be parallel to the ground plane, while walls tend to be perpendicular to this plane. Edges in the scene perpendicular to the ground will appear in each image to be directed towards the vertical vanishing point [16].

If a feature in an image lies on a roof, its appearance in the other image as a function of position along the epipolar line can be predicted if the normal to the ground plane is known. To see why, consider Fig. 2. Suppose the junction P_1P_2 in image1 is given, and our goal is to predict the junction Q_1Q_2 in image2, where the point Q_1 lies anywhere (inside the infinity point) on the epipolar line corresponding to P_1 . For the position Q_1 , the 3-space position of V_1 can be computed as the intersection of the rays through P_1 and Q_1 . This uniquely determines the position of the plane parallel to the ground that contains V_1 . The 3-space positions of the points V_2 and V_3 can now be computed as the intersections of this plane with the rays corresponding to the points P_2 and P_3 , respectively. Finally, the points Q_2 and Q_3 are uniquely determined as central projections of the points V_2 and V_3 , respectively.

Therefore, when an L junction is found in one image, it is initially assumed to arise from a corner of a roof, and its appearance in the other image can be predicted. When an ARROW or FORK junction is found,

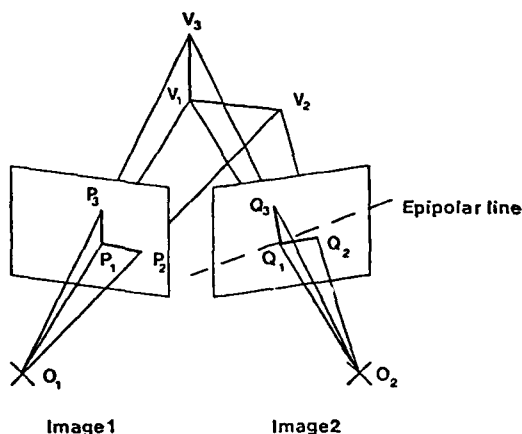


Figure 2: For junction $P_1P_2P_3$, its appearance in image2 can be predicted as a function of position Q_1 along the epipolar line. The normal to plane $V_3V_1V_2$ must be known.

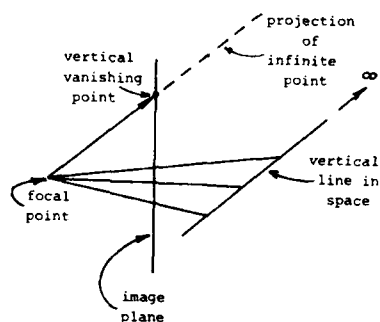


Figure 3: The vector from the focal point to the vertical vanishing point is a 3-space vector in the vertical direction.

the leg of the junction directed towards the vertical vanishing point is initially assumed to arise from a scene edge perpendicular to the ground, while the other two legs are initially assumed to arise from scene edges lying on a roof or on the ground. Again its appearance can be predicted.

Structural relationships between scene vertices are also used to aid in the matching. If two junctions in an image arise from scene vertices at the same height above the ground, the positions of the corresponding junctions in the other image, as a function of position along the epipolar line, can be predicted if the normal to the ground plane is known. This can be shown using similar arguments as before. In Fig. 2, pretend that the points P_1 , Q_1 , and V_1 correspond to positions of separate junctions and vertices. For example, if P_1 and P_3 are two separate junctions in image1, then for some point Q_1 on the epipolar line corresponding to P_1 , the position of the junction Q_3 , corresponding to P_3 , can be predicted if V_1 and V_3 are assumed to lie at the same height. We make the assumption that junctions close to one another in the image often correspond to vertices lying on top of the same building and therefore have approximately the same height.

These matching techniques assume that the vector normal to the ground plane is known. To obtain this vector, we form a vector from the focal point to the vertical vanishing point. As shown in Fig. 3, this results in a 3-space vector in the vertical direction [4], since a line containing the focal point and vertical vanishing point intersects any vertical line at

infinity. The focal length and vertical vanishing point are currently manually obtained.

3.1. Steps in Stereo Analysis

We now provide an example showing how the stereo analysis is performed on the stereo pair of images in Fig. 4. First, linear features are extracted by finding edge points, thinning and linking them, and fitting piecewise linear segments. The resulting line images are shown in Fig. 5. Next, junctions are extracted by placing a 5x5 window around each end point of each line and searching for ends of other lines. Junctions that have been found are labeled in Fig. 5. (See [15] for more details.) Notice that many of the junctions correspond to building corners.

We now want to find potential junction matches between the two images. Let us consider how L junctions are matched. Each L junction is initially assumed to lie on a horizontal scene plane. The shape and orientation of its corresponding junction in the other image, as a function of position along the epipolar line, can therefore be predicted. Each L junction in the first image may therefore usually be matched with several junctions in the second image that have, within tolerance, the predicted shape and orientation. However, we do not try to match only with junctions in the second image that have been previously found. Rather, for every point on the epipolar line (on the appropriate side of the infinity point), a search is made within a pre-specified window for lines that might correspond to the predicted junction. The requirements, however, for two lines to form a junction is more relaxed than the requirements during initial junction search. The matching is performed in two directions, from the first image to the second, and vice versa.

At this point, each junction in one image is associated with a set of potentially matching junctions in the other image. The next step is to find the best of the potential matches, resulting in a single match for each junction. Two criteria are used in determining the best matches:

1. If the image intensities inside two potentially matching junctions are similar, the likelihood that they really match is increased. This is because the two junctions will often have similar intensities if they arise from the same face corner. To measure the degree of similarity, we compute the average intensities of regions along the two legs of the L junction in each image. As depicted in Fig. 6, let A and B be the average intensities of these regions in one image, and let A' and B' be the average intensities of corresponding regions in the other image. Then the degree of similarity, called the *local cost*, is defined as

$$C_{\text{local}} = |A - A'| + |B - B'|.$$

2. As described previously, if two junctions in an image arise from scene vertices that are at the same height, the relative positions of the corresponding junctions in the other image, as a function of position along the epipolar line, can be predicted. We use this to determine whether two sets of junction matches are consistent with one another. Suppose, in Fig. 7, that the junctions J_1 and J_2 in image1 arise from scene vertices that are at the same height. Suppose also that the junction matches (J_1, J'_1) and (J_2, J'_2) have been hypothesized. To measure the degree of consistency between these two sets of matches, we predict the position of the junction in image2 that corresponds to (say) J_2 . Let us refer to the predicted position as J''_2 . If the vector from J'_1 to J''_2 is (a_1, b_1) and the vector from J'_1 to J'_2 is (a_2, b_2) , then the degree of consistency between the two sets of matches, called the *global cost*, is defined as

$$C_{\text{global}} = |a_1 - a_2| + |b_1 - b_2|.$$

To arrive at a unique set of junction matches, the space of potential matches is searched using a beam search [24], which is guided by the above two criteria. The results of this search are displayed in Fig. 8, which shows junctions in one image that have matches in the other image.

Finally, 3D coordinates of vertices and equations of edges are derived using triangulation. Fig. 9 shows a perspective view of the 3D vertices and edges that result. We call this a wire-frame description of the scene.

4. Monocular Analysis

Although stereo is a major source of 3D information, some views of the scene will be only single images. We can also extract 3D information from these images by exploiting task-specific knowledge. We assume that the objects in the scene are trihedral polyhedra containing only vertical and horizontal faces, i.e., faces perpendicular and parallel, respectively, to the ground plane. Our monocular analysis extracts linear structures in the image that represent boundaries of buildings, and then converts these structures into 3D wire frames.

4.1. Steps in Monocular Analysis

This section provides an example showing how the monocular analysis is performed on the image in Fig. 10.

Extracting lines and junctions. The first step is to extract linear segments and junctions from the image. The method used here is the same as that used during stereo analysis (as previously described). First thinned edge points are found, and then lines and junctions are extracted, as shown in Fig. 11.

Locating 2D structures. Next we form linear connected structures in the image by hypothesizing new lines to connect the previously extracted junctions. These connected structures are meant to represent building boundaries and the hypothesized lines are meant to correspond to building edges. The process of hypothesizing connecting lines consists of two steps. First, two junctions may be connected only if a leg of one points at the other, that is, the extended leg meets the other junction. For each pair of junctions that passes this test, a line showing the connection between the two junctions is drawn in Fig. 12.

The second step involves determining which connections shown in Fig. 12 appear as connections in the line image (Fig. 11). For each pair of connected junctions J_i and J_k , we find all segments in the line image that are contained within a thin rectangular window connecting J_i and J_k , and project these segments onto the line connecting the two junctions. Then we consider how much of this line is covered by projected segments. The connection between J_i and J_k is retained only if the percentage of coverage exceeds a threshold. After this pruning step, the junction legs originally extracted in the junction finding step are added, and extraneous legs are deleted. The final connected structures are displayed in Fig. 13.

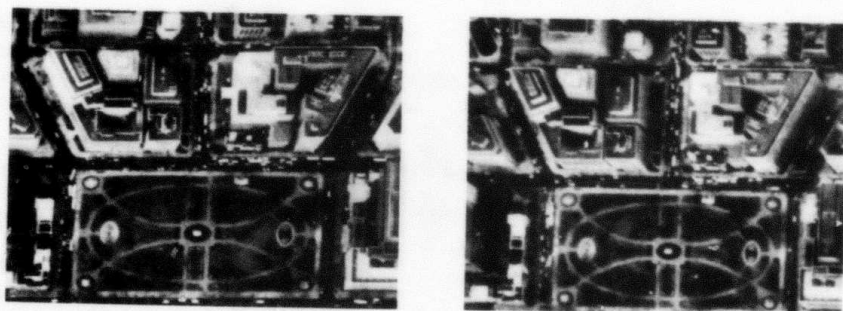


Figure 4: Gray scale stereo images of a region of Washington, D. C.

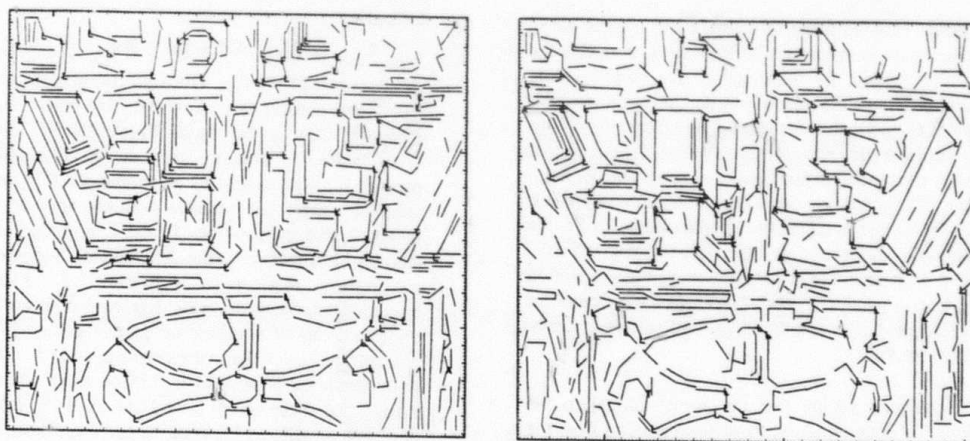


Figure 5: Fitting linear segments after extracting, thinning, and linking edge points. Junctions are classified as L, A (arrow), F (fork), or T.

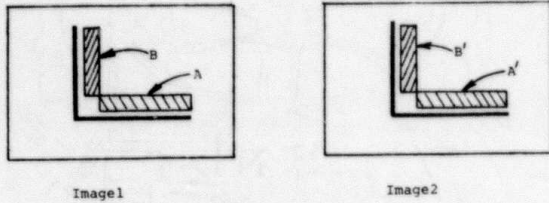


Figure 6: Intensities of corresponding regions of L junctions in the two images are used to compute the local matching cost.

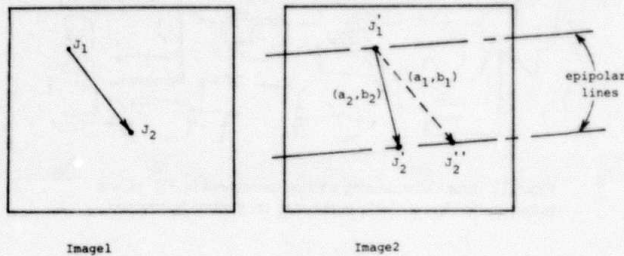


Figure 7: Positional vectors of predicted and actual positions of two junction matches are used to compute the global matching cost.

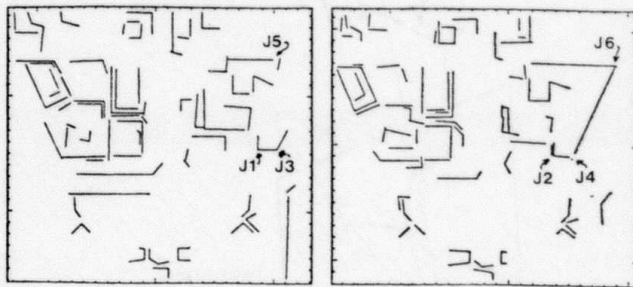


Figure 8: Matches that have been found for the junctions in Fig. 5. Actually, not all matches are correct. For example, although the junction matches (J1,J2) and (J3,J4) are correct, the match (J5,J6) is incorrect.

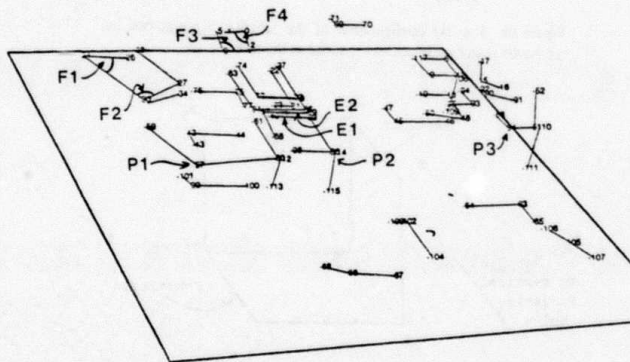


Figure 9: Perspective view of 3D wire-frame description derived from a set of matches similar to those in Fig. 8.

Obtaining 3D wire frames. The next step is to convert the 2D structures into 3D wire frames. First, the lines that form the 2D structures are labeled as either "vertical" or "horizontal" depending on whether or not they are directed toward the vertical vanishing point [16]. Next, we use the position of the vertical vanishing point to calculate the vector in the vertical direction, as described in an earlier section. Let us now consider how to recover the 3D configuration of the junction $p_1 p_2 p_3 p_4$ in Fig. 14.

Suppose that line $p_2 p_4$ has been labeled "vertical" and lines $p_1 p_2$ and $p_3 p_4$ have been labeled "horizontal". Let u be the unit vector in the vertical direction. This vector is normal to all horizontal planes. First we would like to determine the 3-space position of v_2 , corresponding to the junction point p_2 . Since it is impossible to determine the actual position of this point from a single image without special information, the position is determined as some arbitrary point lying on the ray through p_2 , i.e., the depth a of v_2 is arbitrarily chosen. The horizontal plane $v_1 v_2 v_3$ can now be established, since it contains v_2 and its normal vector is u . The 3-space positions of the points v_1 and v_3 can then be computed as the intersections of this plane with the rays through p_1 and p_3 , respectively. Finally, the 3-space position of the point v_4 is computed as the intersection of the ray through p_4 with the line through v_2 along the vector u .

Although this technique permits us to recover the 3D configuration of any junction relative to some arbitrary depth, it is not useful to apply it directly to the junctions in the original line image (Fig. 11) because the relative heights above the ground plane of the corresponding vertices cannot be determined; the height of each vertex is arbitrarily chosen without relation to the heights of other vertices. It is more useful, however, to apply the technique to the 2D structures in Fig. 13, since the heights of the vertices within each structure can be related. To see how this is done, consider the example in Fig. 15, which shows a 2D structure. Suppose lines $p_1 p_6$ and $p_3 p_4$ have been labeled "vertical", while the other solid lines have been labeled "horizontal". Applying our technique to (say) point p_1 , the 3-space positions of the vertices corresponding to points p_1 , p_2 , and p_6 can be determined relative to some arbitrary depth a for p_1 . If the technique is applied next to point p_2 , the 3-space position of point p_3 can be determined as a function of the depth a . This procedure continues with points p_6 , p_4 , and so on, until the 3D configuration of the whole structure has been determined, relative to some arbitrary depth.

In order to obtain a coherent scene description, the depths of the different structures in the scene must be related. We use two methods to do this. The first method involves finding structures that lie on the ground plane. Suppose a junction point p of such a structure is hypothesized to arise from a vertex lying on the ground. Then the 3-space position of the vertex may be obtained as the intersection of the ground plane with the ray through p . The normal vector u to the ground plane is known, but the distance d from the focal point to the ground plane is arbitrarily chosen. Since the 3-space position of all junctions arising from ground points can be calculated in this manner, the depths of all structures containing such points can be related to one another through the parameter d .

To hypothesize junctions that arise from vertices lying on the ground plane, we use the observation that if a line labeled "vertical" connects two junctions (e.g., line $p_1 p_6$ in Fig. 15), the line is directed toward the vertical vanishing point with respect to one junction, but away from this vanishing point with respect to the other junction. The latter junction is assumed to represent a vertex lying on the ground plane. Points p_1 and p_3 in Fig. 15 are examples of such junctions. The 3-space positions of these junctions are then calculated, and their values are propagated throughout their structures as described previously.

Figure 10: Aerial photograph showing part of Washington, D.C. This is a different view of the same scene as in Fig. 4.

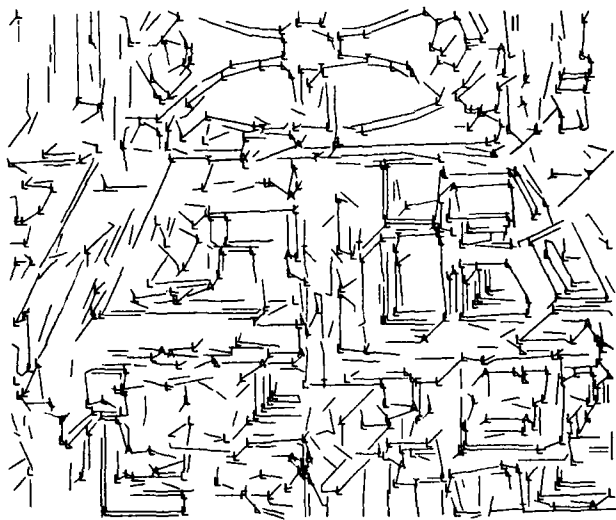


Figure 11: Lines fitted to edge points extracted from Fig. 10. Junctions in the image are classified as L, A (arrow), F (fork), or T.

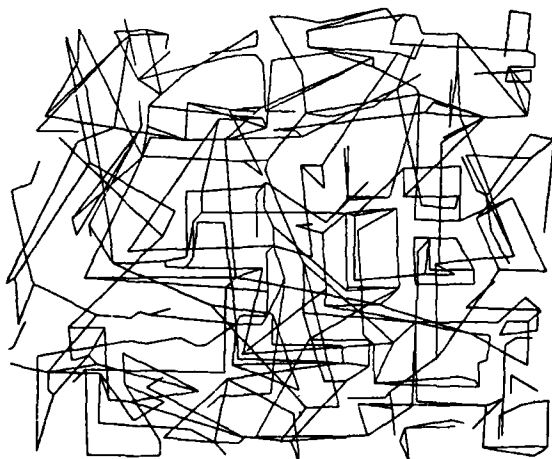


Figure 12: Each line represents a possible connection between the junctions at its two end points. Each end point corresponds to a junction in Fig. 11.



Figure 13: Result after pruning junction connections in Fig. 12, and adding junction legs originally extracted in the junction finding step.

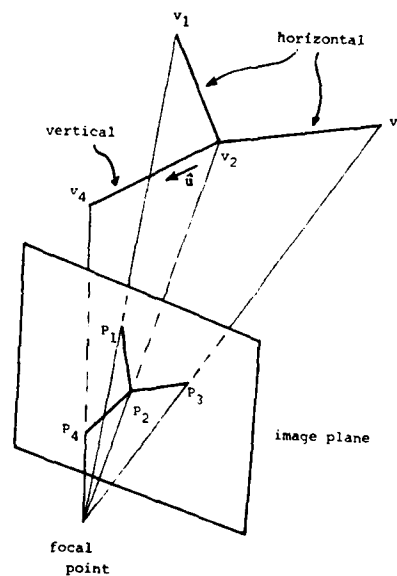


Figure 14: The 3D configuration of the junction $p_1 p_2 p_3 p_4$ can be recovered under assumptions explained in the text.

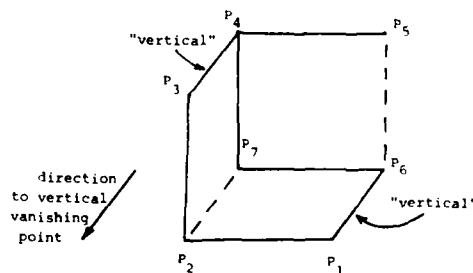


Figure 15: The solid lines represent a connected 2D structure. The dashed lines are for the reader's convenience to make the 3D shape more apparent.

There are many structures in Fig. 13 that do not contain points lying on the ground plane. Nevertheless, the heights of some of these structures can be determined using the rule that if two lines are aligned in the image, they are often aligned in 3-space. Suppose, in Fig. 16, that points p_1 through p_7 have already been assigned 3D coordinates, and we want to obtain the 3-space position of the 2D structure $p_8p_9p_{10}p_{11}$. Since the lines p_8p_9 and p_8p_{11} are aligned in the image and both are labeled "horizontal", they are assumed to be aligned in the scene and to lie in the same horizontal plane. The 3-space position of (say) point p_8 is therefore determined as the intersection of this plane with the ray through p_3 . The 3D coordinates of this point may then be propagated to points p_9 , p_{10} , and p_{11} , as described previously. Note that all 3D positions are functions of the parameter d , which is arbitrarily chosen for the equation of the ground plane.

Fig. 17 depicts a perspective view of the 3D wire frames obtained using these methods.

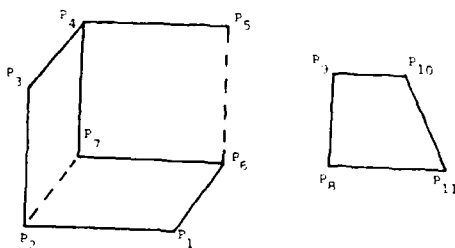


Figure 16: If the 3D configuration of the structure on the left has been determined, the relative 3D position of the structure on the right may also be determined because lines p_8p_9 and p_8p_{11} are aligned.

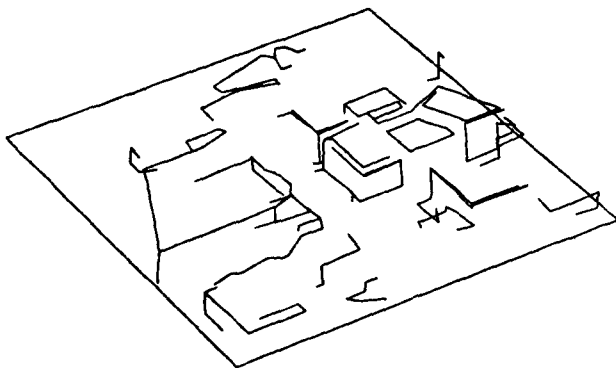


Figure 17: Perspective view of 3D wire frames generated from Fig. 13.

5. Representing and Manipulating the 3D Scene Model

The representation we have developed for the 3D scene model draws on ideas from geometric modelling used in computer-aided design systems [1, 23]. In these systems, however, the 3D models are usually derived through interaction with a user. Our case is different in that (1) the 3D models are derived automatically from 2D images, and (2) many portions of the scene are unknown or recovered with errors because of occlusions or unreliable analysis.

The following factors have determined how the scene model is represented and manipulated.

1. Partially complete, planar-faced objects must be efficiently described by the model. It is therefore represented as a graph in terms of symbolic primitives such as faces, edges, vertices, and their topology and geometry. Information is added and deleted by means of these primitives.
2. The model must be easy to use in matching.
3. Because scene approximations are often more useful if they contain reasonable hypotheses for parts of the scene for which there are partial data, we introduce mechanisms that permit hypotheses to be generated, added, and deleted.
4. Because incremental modifications to the model must be easy to perform, we introduce mechanisms to (a) add primitives to the model in a manner such that constraints on geometry imposed by these additions are propagated throughout the model, and (b) modify and delete primitives if discrepancies arise between newly derived and current information.

The 3D structure in the scene is represented in the form of a graph, called the *structure graph*. The nodes and links represent primitive topological and geometric constraints. The structure graph is incrementally constructed through the addition of these constraints. As constraints are accumulated, their effects are propagated to other parts of the graph so as to obtain globally consistent interpretations.

Nodes in the structure graph represent either primitive topological elements (i.e., faces, edges, vertices, objects, and edge-groups (which are rings of edges on faces)) or primitive geometric elements (i.e., planes, lines, and points). Face, edge, and vertex nodes are tagged as either *confirmed* or *unconfirmed*. Confirmed means that the element represented by the node has been derived directly from images. Unconfirmed means that the element has only been hypothesized.

The primitive geometric elements serve to constrain the 3-space locations of faces, edges, and vertices. Plane and line nodes contain plane and line equations, respectively. Point nodes contain coordinate values. The structure graph contains two types of links: the *part-of* link, representing the part/whole relation between two topological nodes, and the *geometric constraint* link, representing the constraint relation between a geometric and topological node.

Fig. 18 shows a simple example of a structure graph consisting of two objects, *obl* and *ob2*. Arrows with single lines represent part-of links, and arrows with double lines represent geometric constraint links. The faces are represented as f_i , the edge-groups as g_i , the edges as e_i , and the vertices as v_i . The graph shows one point node pt and one plane node pl . Further details on representing and manipulating the 3D scene model may be found in [15, 14].

6. Generating the 3D Scene Model

The result of image analysis is a 3D wire-frame description that represents 3D vertices and edges which correspond to portions of boundaries of objects in the scene. We construct a surface-based description -- the 3D scene model -- from these boundaries by hypothesizing new vertices, edges, and faces using task-specific knowledge. Some of the rules used here will be described next, and will be illustrated on the wire frames in Fig. 9.

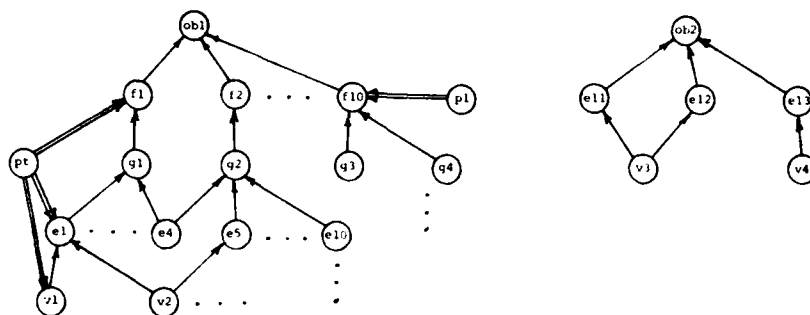


Figure 18: Simple example of a structure graph consisting of two objects, *ob1* and *ob2*.

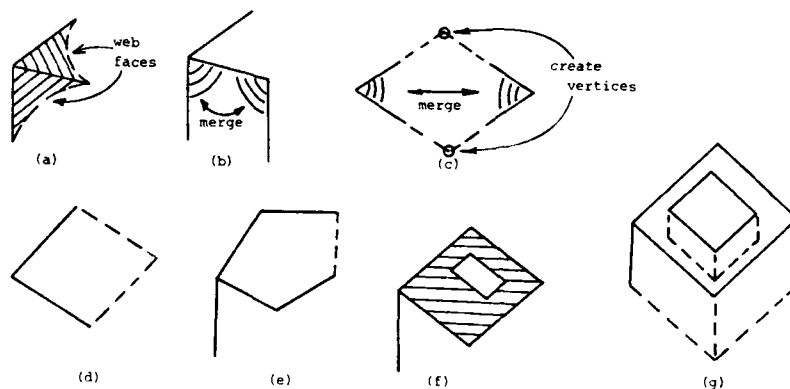


Figure 19: Obtaining a surface-based description from wire frames.

Each adjacent pair of legs ordered around a wire-frame vertex is assumed to correspond to the corner of a planar face. A partial face, called a *web face*, is generated for each such pair (Fig. 19a). Next, web faces that represent corners of a single face are merged. Web faces may either be touching (e.g., Fig. 19b, and F1 and F2 in Fig. 9) or non-touching (e.g., Fig. 19c, and F3 and F4 in Fig. 9). When merging two non-touching faces, the two edges on which each matching pair of end points lie are extended in space and intersected. The intersection points form two new vertices on the resulting face.

Incomplete faces are then completed either as parallelograms (for faces consisting of a single corner (Fig. 19d)) or as polygons (for faces containing three or more connected edges (Fig. 19e)). Next, one face is assumed to represent a hole in another face if (1) the planes of the faces are nearly parallel and close to each other, and (2) the boundary of the first face, when projected onto the plane of the second face, falls inside the boundary of that face (Fig. 19f).

At this point, many objects will be only partially complete because they are not closed. Since we are dealing with urban scenes, faces that lie high enough above the ground are assumed to represent roofs of buildings. A hypothesized vertical wall is dropped towards the ground from each edge of such faces, unless the edge is already part of another face (Fig. 19g). Each wall is dropped either to the ground or to the first face it intersects on the way down.

Fig. 20 shows perspective views of the resulting scene model. Notice that one of the buildings has a hole in it, through the roof. The planar patches at the "front" of the scene are part of the ground. Fig. 21 shows the scene model generated when these techniques are applied to the wire-frame description obtained using monocular analysis (Fig. 17).

In order to render more realistic displays, gray scale is added to them [10]. This is useful for realistically simulating the appearance of the scene from arbitrary viewpoints. We associate with each face in the model an intensity patch obtained from the image. For faces that are partially occluded in the image, the intensity patch is associated with the visible portions. Figs. 22 and 23 show the results of adding gray scale to the faces of the models in Figs. 20 and 21, respectively.

7. Combining New Views with Current Model

The process of incorporating a 3D wire-frame description extracted from a new view into the current scene model can be divided into three main steps:

1. The wire-frame data must first be matched to the current model. This process provides (a) the scale transformation and coordinate transformation from the wire-frame data to the model, and (b) corresponding elements (i.e., vertices and edges) in the two.

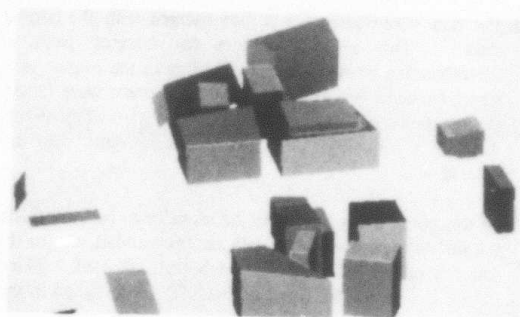
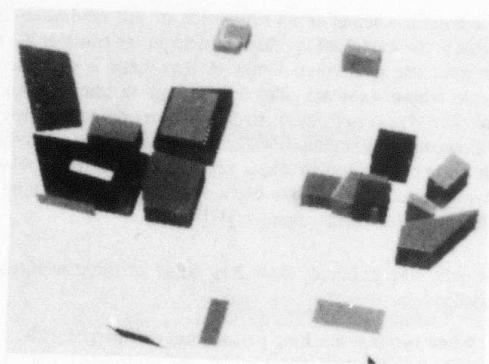


Figure 20: Perspective views of buildings reconstructed from wire-frame data in Fig. 9.

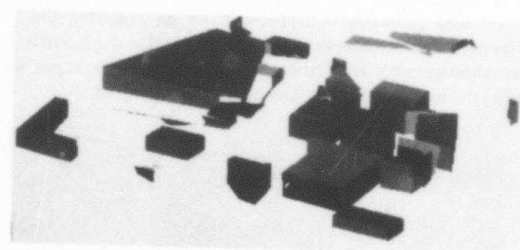
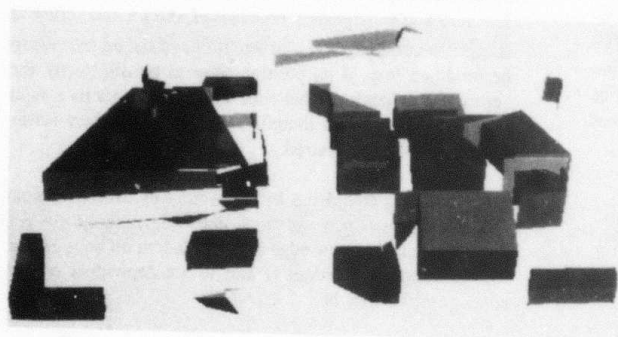


Figure 21: Perspective views of buildings reconstructed from wire-frame data in Fig. 17.

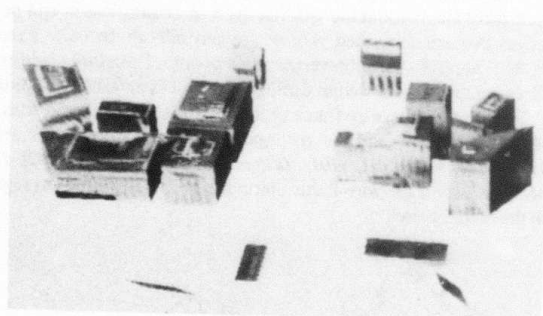


Figure 22: Reconstructed buildings of Fig. 20 with gray scale, derived from the left image in Fig. 4, mapped onto faces.

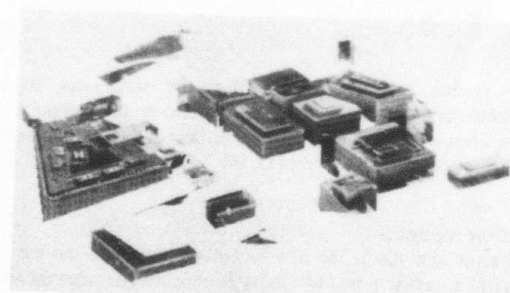


Figure 23: Reconstructed buildings of Fig. 21 with gray scale, derived from Fig. 10, mapped onto faces.

2. The new wire-frame data is then merged with the current model. This process includes (a) merging pairs of corresponding elements, and (b) adding to the model wire-frame elements for which no correspondences were found. During the merging process, hypothesized parts of the model that are inconsistent with the new wire-frame data are deleted.

3. At this point, many objects in the model may be incomplete because (a) new wire-frame data has been added, and/or (b) some hypothesized elements have been deleted. These objects are completed using the techniques described in the previous section.

To see how these steps are carried out, consider the example of incorporating the information from a second view into the scene model of Fig. 20. This scene model was constructed from the set of wire frames (Fig. 9) automatically extracted from a "front" view of the scene (Fig. 4). The second set of wire frames, shown in Fig. 24, was manually generated to simulate information available from an opposing point of view (viewing the scene from the "back"). Notice that the information in Fig. 9 emphasizes edges and vertices facing the front of the scene, while those facing the back of the scene are emphasized in Fig. 24.

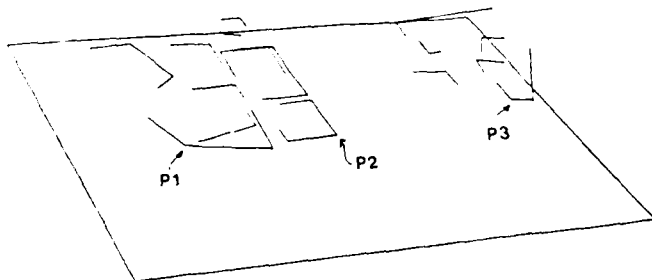


Figure 24: Perspective view of manually generated vertices and edges. The viewpoint for this drawing is chosen to be similar to Fig. 9. Points P1, P2, and P3, for example, correspond to points P1, P2, and P3 in Fig. 9.

We assume in this example that the scale and coordinate transformations from the new wire-frame data to the current model is known. Next, corresponding edges and vertices in the data and model are obtained, as described elsewhere [15, 14].

7.1. Discrepancies

We must now merge the new wire-frame data into the model. An important issue here is how to handle discrepancies between the two. We consider the following two types of discrepancies:

1. After the coordinate system of the wire-frame data has been transformed to that of the model and scale adjustments have been made, corresponding pairs of confirmed vertices and edges may not register perfectly in 3-space. In order to merge them into single elements, we perform a "weighted averaging" of their positions.
2. Hypothesized elements in the model may be inconsistent with newly obtained elements. We handle this by deleting such hypothesized elements.

To determine whether or not hypotheses are still valid when confirmed elements in the model are modified or deleted, we consider the elements which gave rise to the hypotheses. A hypothesis is dependent on all elements whose existence directly resulted in the creation of the hypothesis. If one of these elements is modified or deleted, the hypothesis must also be modified or deleted since the conditions under which it was created are no longer valid. The dependency relationships for hypothesized elements are explicitly recorded at the time of their creation using dependency pointers [11].

The following examples show how some of these relationships are recorded:

1. When two non-touching partial faces are merged, (Fig. 25a) each face has two edges which are intersected with their counterparts in the other face. The intersection points form two new hypothesized vertices, each of which is dependent on the two edges whose intersection gave rise to it. In Fig. 25a, vertex $v1$ is dependent on edges $e1$ and $e3$, and vertex $v2$ is dependent on edges $e2$ and $e4$. If one of the edges were to be modified (e.g., if its position were to be displaced), the vertex that depends on that edge would no longer be a valid hypothesis, and would therefore be deleted. A new vertex might then be hypothesized.
2. When a face is completed by connecting its two end points (Fig. 25b), two new vertices and one new edge are hypothesized. The new edge $e4$ is dependent on both $e1$ and $e3$, while the new vertices $v1$ and $v2$ are dependent on the edges on which they lie.

When a confirmed edge or vertex in the model is modified or deleted, the set of all hypothesized elements that depend on it are deleted. Recursively, elements depending on deleted ones are also deleted.

7.2. Merging

The procedure that merges corresponding wire-frame and model objects takes into account the fact that the 3-space positions of end points of edges that are confirmed vertices are generally much more accurate than the positions of non-vertex end points. Therefore, confirmed vertices are given more weight during merging. As an example, consider Fig. 26. Suppose the wire-frame object in (b) is to be merged with the model object in (a), and the corresponding edges and vertices are as follows: $(v2, v100)$, $(v3, v101)$, $(e2, e100)$, $(e3, e101)$, $(e4, e102)$, $(e12, e104)$. We assume the wire-frame object has been transformed to register with the model object.

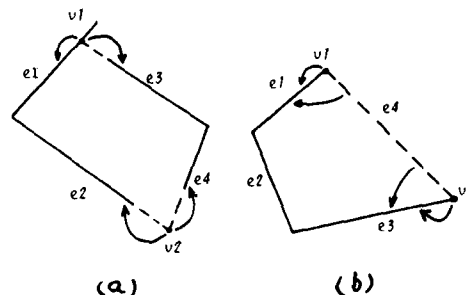


Figure 25: Generating dependencies for hypothesized edges and vertices. The dependence of an element on another is depicted as an arrow from the former to the latter. (a) Two non-touching partial faces are merged. (b) A face is completed.

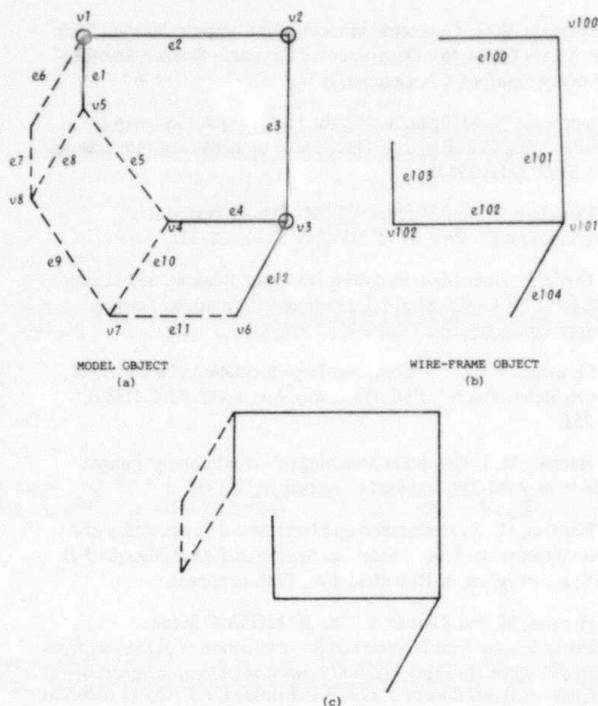


Figure 26: The wire-frame object in (b) is to be merged with the model object in (a). The confirmed edges of the model object (indicated by solid lines) are $e1$, $e2$, $e3$, $e4$, and $e12$; the confirmed vertices (indicated by circles) are $v1$, $v2$, and $v3$. Dashed lines represent hypothesized edges. (c) The result after merging.

The merging procedure starts by merging corresponding vertices. Pairs of vertices ($v2$, $v100$) and ($v3$, $v101$) in Fig. 26) are combined into single vertices with coordinates of the midpoint between them. At this point, all corresponding pairs of edges will share at least one vertex. The corresponding edges are merged next as follows:

1. If the two edges share both their vertices ($(e3, e101)$ in Fig. 26), the new edge connects the two new vertices already generated.
2. If one edge has two confirmed vertices but the other does not ($(e2, e100)$ and $(e4, e102)$ in Fig. 26), the new edge is the same as the former. Notice that the non-vertex end point in this case is given zero weight.
3. If the two edges share one vertex and the other end points are not confirmed ($(e12, e104)$ in Fig. 26), the new edge is the "average" of the two edges.

If a model edge to be merged contains only one confirmed vertex (e.g., $e4$ and $e12$ in Fig. 26), then all hypothesized elements that recursively depend on this edge are deleted. For example, the hypothesized elements that recursively depend on $e4$ in Fig. 26 are the vertices $v4$ and $v7$, and the edges $e5$, $e10$, $e9$, and $e11$.

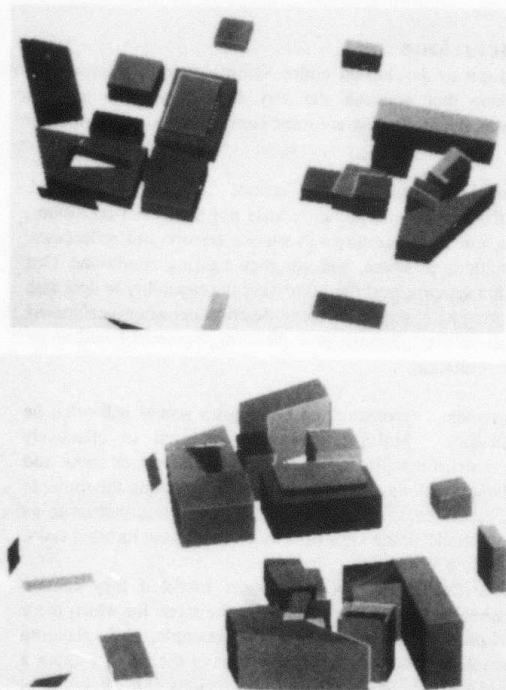


Figure 27: Perspective views of buildings derived by incorporating the wire-frame data in Fig. 24 into the model in Fig. 20.

After all corresponding elements of the two objects have been merged, the edges and vertices remaining in the wire-frame object that were not merged ($e103$ in Fig. 26) are added to the model object. The final configuration after merging is shown in Fig. 26c. This object is incomplete and must be completed using the techniques described in an earlier section.

7.3. Results of Merging

When these procedures are applied to the wire-frame data in Fig. 24 and the scene model in Fig. 20, we obtain the updated scene model shown in Fig. 27. The updated version has two important improvements over the initial version. First, the updated model contains more buildings since new wire-frame data, some of which represent new buildings, have been incorporated into the initial model. Second, for many buildings described in both versions of the model, the positions of vertices and edges are more accurate in the updated version. This is because many hypothesized vertices and edges are replaced by accurate ones obtained from the new data, and many confirmed vertices and edges are merged with corresponding ones in the data by "averaging" their positions, generally decreasing the amount of error.

The shape of the large hole in the roof of one of the buildings has changed from a rectangle in the initial model to an almost triangular quadrilateral in the updated version. When compared with the source images in Fig. 4, the rectangular shape would seem more accurate. However, the positions of the edges and vertices that form the hole are more accurate in the updated model in the sense that they are more faithful to the wire-frame descriptions derived from the images.

This experiment demonstrates how information provided by each additional view allows the model to be incrementally made more complete and accurate.

8. Conclusions

We set out to develop an entire vision system to interpret complex images, one that goes all the way from images to symbolic 3D descriptions. The following are some conclusions we can draw from this project.

1. Complex images usually cannot be fully interpreted. Difficulties in interpretation arise not only from occlusions, but also from variations in surface texture and reflectance, variations in shape, and complex lighting conditions. Our vision systems must therefore have the capability to deal with approximate, imperfect scene descriptions when performing tasks such as matching, path planning, or model-based image interpretation.
2. Incremental reconstruction of complex scenes will often be necessary. Multiple views are required to effectively reconstruct complex scenes. A system that moves about and interacts with its environment in order to obtain the multiple views will be able to gradually add more information to its scene model at the same time that it carries out its other tasks.
3. Scene descriptions are often more useful if they contain reasonable hypotheses for parts of the scene for which there are only partial or no data. For example, path planning cannot be done for occluded regions of the scene without a good guess about what lies in these regions. If the hypotheses turn out to be incorrect, they should eventually be modified. Our vision systems must therefore have mechanisms for intelligently generating hypotheses, verifying them, and modifying them.
4. Task-specific knowledge is very useful at all levels of complex image interpretation, from low-level image analysis to high-level formation of symbolic descriptions. Knowledge of block-shaped objects in an urban scene is used in the 3D Mosaic system for stereo analysis, monocular analysis, and reconstructing shapes from the wire frames.
5. Stereo matching of 2D structural features (such as junctions) may be important for complex images and should be further investigated.

References

1. Baer, A., Eastman, C., and Henrion, M. "Geometric Modelling: a Survey." *Computer-Aided Design* 11 (September 1979), 253-272.
2. Baker, H. H. "Three-Dimensional Modelling." *Proc. IJCAI-77* (August 1977), 649-655.
3. Baker, H. H., and Binford, T. O. "Depth from Edge and Intensity Based Stereo." *Proc. IJCAI-81* (1981), 631-636.
4. Barnard, S. T. "Methods for Interpreting Perspective Images." *Proc. ARPA Image Understanding Workshop* (September 1982).
5. Barnard, S. T. and Fischler, M. A. "Computational Stereo." *Computing Surveys* 14, 4 (December 1982).
6. Barnard, S. T. and Thompson, W. B. "Disparity Analysis of Images." *IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-2*, 4 (July 1980), 333-340.
7. Barrow, H. G., Bolles, R. C., Garvey, T. D., Kremers, J. H., Tenenbaum, J. M., and Wolf, H. C. "Experiments in Map-guided Photo Interpretation." *Proc. IJCAI-77* (August 1977), 696.
8. Baumgart, B. G. Geometric Modeling for Computer Vision. Tech. Rept. STAN-CS-74-463, Department of Computer Science, Stanford University, Stanford, CA, October, 1974.
9. Bourne, D. A., Milligan, R., Wright, P. K. "Fault Detection in Manufacturing Cells Based on Three Dimensional Visual Information." *Proc. SPIE* (May 1982).
10. Devich, R. N., and Weinhaus, F. M. "Image Perspective Transformations." *Proc. SPIE* 238 (July 1980), 322-332.
11. Doyle, J. Three Short Essays on Decisions, Reasons, and Logics. Tech. Rept. STAN-CS-81-864, Department of Computer Science, Stanford University, Stanford, CA, May, 1981.
12. Grimson, W.E.L. "A Computer Implementation of a Theory of Human Stereo Vision." *Phil. Trans. Roy. Soc. Lond. B292* (1980), 217-253.
13. Hannah, M. J. Computer Matching of Areas in Stereo Images. Tech. Rept. AIM-239, Stanford University, July, 1974.
14. Herman, M. Representation and Incremental Construction of a Three-Dimensional Scene Model. In *Sensors and Algorithms for 3-D Machine Perception*, A. Rosenfeld, Ed., 1984, to appear.
15. Herman, M. and Kanade, T. The 3D MOSAIC Scene Understanding System: Incremental Reconstruction of 3D Scenes from Complex Images. In *From Pixels to Predicates: Recent Advances in Computational and Robotic Vision*, A.P. Pentland, Ed., Ablex Publishing Company, 1984, to appear.
16. Kender, J. R. "Environmental Labelings in Low-Level Image Understanding." *Proc. IJCAI-83* (August 1983), 1104-1107.
17. Lucas, B. D., and Kanade, T. "An Iterative Image Registration Technique With an Application to Stereo Vision." *Proc. IJCAI-81* (August 1981), 674-679.
18. Martin, W. N. and Aggarwal, J. K. "Volumetric Descriptions of Objects from Multiple Views." *IEEE Trans. on Pattern Analysis and Machine Intelligence PAMI-5*, 2 (March 1983), 150-158.
19. McKeown, D. M. MAPS: The Organization of a Spatial Database System Using Imagery, Terrain, and Map Data. Tech. Rept. CMU-CS-83-136, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, July, 1983.
20. Moravec, H.P. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. Tech. Rept. CMU-RI-TR-3, The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, September, 1980.
21. Ohta, Y., and Kanade, T. Stereo by Intra- and Inter-scanline Search Using Dynamic Programming. Tech. Rept. CMU-CS-83-162, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, October, 1983.
22. Potmesil, M. "Generating Models of Solid Objects by Matching 3D Surface Segments." *Proc. IJCAI-83* (August 1983), 1089-1093.
23. Requicha, A. A. G. "Representations for Rigid Solids: Theory, Methods, and Systems." *Computing Surveys* 12, 4 (December 1980), 437-464.
24. Rubin, S. "Natural Scene Recognition Using Locus Search." *Computer Graphics and Image Processing* 13 (1980), 298-333.
25. Underwood, S. A. and Coates, C. L. "Visual Learning from Multiple Views." *IEEE Transactions on Computers*, 6 (June 1975), 651-661.

Hierarchical Warp Stereo

Lynn H. Quam
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

September 10, 1984

Abstract

This paper describes a new technique for use in the automatic production of digital terrain models from stereo pairs of aerial images. This technique employs a coarse-to-fine hierarchical control structure both for global constraint propagation and for efficiency. By the use of disparity estimates from coarser levels of the hierarchy, one of the images is geometrically warped to improve the performance of the cross-correlation-based matching operator. A newly developed surface interpolation algorithm is used to fill holes wherever the matching operator fails. Experimental results for the Phoenix Mountain Park data set are presented and compared with those obtained by ETL.

1 Introduction

The primary objective of this research was to explore new approaches to automated stereo compilation for producing digital terrain models from stereo pairs of aerial images. This paper presents an overview of the hierarchical warp stereo (HWS) approach, and shows experimental results when it is applied to the ETL Phoenix Mountain Park data set.

The stereo images are assumed to be typical aerial-mapping pairs, such as those used by USGS and DMA. Such pairs of images are different perspective views of a 3-D surface acquired at approximately the same time and illumination angles. Normally these views are taken with the camera looking straight downward. The major effect of non verticality is to increase the incidence of occlusion, which increases the difficulty of point correspondence.

We shall call one of these images the "reference image," and the other the "target image." We will be searching in the target image for the point that best matches a specified point in the reference image.

It is also assumed that the epipolar model for the stereo pair is known, which means that for any given point in one image we can determine a line segment in the other image that must contain the point, unless it is occluded from view by other points on the 3-D surface. This is certainly a reasonable assumption, since an approximation to the epipolar model can be derived from a relatively small number of point correspondences if the parameters of the imaging platform are not known a priori.

The primary goal is to automatically determine correspondences between points in the two images, subject to the following criteria:

- Minimize the rms difference between the disparity measurements and "ground truth." Without ground truth, we cannot measure this.
- Maximize the sensitivity of the disparity measurements to small-scale terrain features, while minimizing the effects of noise.
- Minimize the frequency of false matches.
- Minimize the frequency of match failures.

These criteria are mutually exclusive. Under ideal conditions, increasing the size of the match operator decreases the effects of noise on the disparity measurement, but it also diminishes sensitivity to small terrain features. Similarly, tightening the match acceptance criteria reduces the frequency of false matches, but results in more frequent match failures.

One of the goals of this system is to minimize the number of parameters that must be adjusted individually for each stereo pair to get optimum performance.

2 Approach

This section briefly explains the HWS approach, which consists of three major components:

- Coarse-to-fine hierarchical control structure for global constraint propagation as well as for efficiency.
- Disparity surface interpolation to fill holes wherever the matching operator fails.
- Geometric warping of the target image by using disparity estimates from coarser levels of the hierarchy to improve the performance of the cross-correlation-based matching operator.

2.1 The Use of Hierarchy and Surface Interpolation to Propagate Global Constraints

The goal of stereo correspondence is to find the point in the target image that corresponds to the same 3-D surface point as a given point in the reference image. It is often impossible to select the correct match point with only the image information that is local to the given point in the reference image in combination with the image information along the epipolar line segment in the target image. When the 3-D surface contains a replicated pattern, there is the likelihood of match point ambiguity. Let us consider, for example, a stereo pair that contains a parking lot

This research was supported by the Defense Advanced Research Projects Agency under Contract No. MDA 903-83-C-0027.

with repetitive markings delimiting the parking spaces. Around the edges of the lot there are image points that can be matched unambiguously. Within the parking lot, ambiguity is likely, depending on the orientation of the repetitive patterns with the epipolar line. A successful stereo correspondence system must be able to use global match information to resolve local match-point ambiguity.

HWS approaches this problem in two ways. First, global constraints on matches are propagated by the coarse-to-fine progression of the matching process. Disparities computed at lower resolution are employed to constrain the search in the target image to a small region of the epipolar line, which also greatly reduces the probability of selecting the wrong point when ambiguity is present. Second, whenever the match process fails to find a suitable match or detects a possible match ambiguity, a disparity estimate is inserted that is based on a surface interpolation algorithm, which uses information from a neighborhood around the disparity "hole," with the size of the neighborhood depending on the number of neighboring "holes."

2.2 The Use of Image Warping to Improve Correlation Operator Performance

One of the greatest problems in the use of area correlation for match point determination is the distortion that occurs because of disparity changes within the correlation window. Since area-based correlation matches areas, rather than individual points, the disparity it calculates is influenced by the disparities of all of the points in the window, not just the point at the center. When there are high disparity gradients or disparity discontinuities, the correlation calculated for the correct disparity can actually be so poor that some other disparity will have a higher correlation score.

The effect of correlation window distortion can be greatly mitigated in a hierarchical system by using the disparity estimates from the previous level of matching to warp the target image geometrically at its current resolution level into closer correspondence with the reference image.

2.3 Related Work

Norvelle [1] implemented a semi automatic stereo compilation system at the U.S. Army Engineer Topographic Laboratories (ETL) that operates in a single pass through the images. It uses disparity surface extrapolation both to predict the region of the epipolar segment for matching and to estimate the local surface orientation so as to warp the correlation window. He found that these techniques improved the performance of the system significantly, but that considerable manual intervention was needed when the surface extrapolator made bad predictions, or when the image contained areas with no information for matching, with ambiguities, or with occlusions.

3 Sequence Of Operations In Hierarchical Warp Stereo

Figure 1 illustrates the hierarchical control structure of the system.

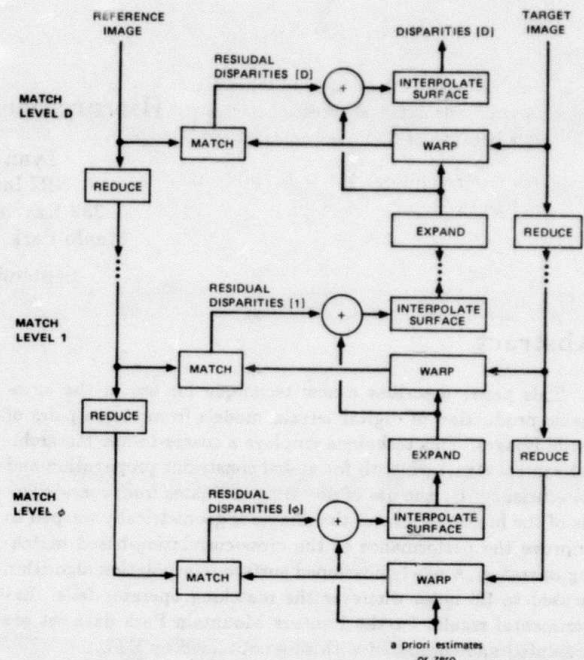


FIGURE 1

Block Diagram of Hierarchical Control Structure

1. Initialize:

- Start with a stereo pair of images (assumed to be of the same dimensions).
- Call one of these images the "reference image," the other the "target image."
- Construct Gaussian pyramids (Burt [2]) $reference_i$ and $target_i$ for each image. The images at level i in these pyramids correspond to reductions of the original images by a factor of 2^i .
- Set $disp_{-1}$ to either the a priori disparity estimates or all zeros.
- Start the iteration at level $i = 0$.
- Choose the pyramid depth D so that:

$$D = \text{ceiling}(\log_2(\text{uncertainty})) - 1.$$

where *uncertainty* is an estimate of the maximum difference between $disp_{-1}$ and the "true" disparities. This guarantees the "true" disparities will be within the range $(-2 : +2)$ at level 0 of the matching.

2. Warp: Use the disparity estimates $2 * disp_{i-1}$ to warp $target_{D-i}$ geometrically into approximate alignment with $reference_{D-i}$. Note that the factor of two is equal to the ratio of image scales between level i and level $i - 1$ of the hierarchy.
3. Match: Using the matching operator, compute the residual disparities $\Delta disp_i$ between the warped target and the reference images at level i .

4. Refine: Compute the refined disparity estimates:

$$disp_i = 2 * disp_{i-1} + \Delta disp_i.$$

5. Fill: Use the surface interpolation algorithm to fill in disparities estimates at positions where matching operator fails because of no image contrast, ambiguity, etc.
6. Increase resolution: If $i = D$, quit; otherwise let $i = i + 1$ and go to Step 2.

4 Disparity Estimation

Disparity estimation consists of three parts:

- Computing match operator scores for disparities along an epipolar segment.
- Accepting or rejecting the collection of scores according to a model for the shape of the correlation peak.
- Estimating the subpixel disparities at acceptable peaks.

4.1 Match Score Operator

The HWS approach presented here can be implemented with a variety of match operators. All results reported here were obtained with an operator that closely approximates Gaussian-weighted normalized cross correlation. The values of the Gaussian weights decrease with Euclidean distance from the center of a square correlation window. In the examples shown here, the window dimension is 13×13 pixels with a standard deviation of approximately 2 pixels in the Gaussian weights. Preliminary results indicate that the Gaussian-weighted correlation operator is better than uniformly weighted correlation operators at locating changes in disparity while maintaining a given level of disparity precision.

4.2 Evaluation of Correlation Surface Shape

The match operator reports a failure if any of the following conditions exist:

- Disparity out of range: The maximum match score is found at either extreme of the epi-polar segment.
- Multiple peaks: The best and next best match scores is found at disparities that differ by more than one pixel.

There are other models for the expected shape of the correlation surface that can be based on the autocorrelation surface shape of the windows in the reference and target images. Further investigation is needed to evaluate the utility of such models for both surface shape evaluation and disparity estimation.

4.3 Subpixel Disparity Estimation

The subpixel location of the correlation surface peak is estimated by parabolic interpolation of both the x and y directions of disparity. For each direction, three adjacent match scores – s_{i-1} , s_i , and s_{i+1} , where s_i is the maximum score – are used to compute the peak as follows:

$$.5 * \frac{s_{i+1} - s_{i-1}}{2 * s_i - s_{i+1} - s_{i-1}}$$

More complicated approaches to peak estimation, such as two-dimensional least-squares fitting of the correlation surface, might yield better estimates, but at a higher computational cost.

5 Surface Interpolation Algorithm

The goal of the surface interpolation algorithm is to estimate values for the disparity surface at points where the match operator reported failure; such points will be called "holes." The approach to filling a hole at location x, y is to model the surface by employing the disparity measurements over the set of non-holes \bar{H} in the $n \times n$ pixel neighborhood centered at x, y . The set H contains the indices of all holes in the neighborhood.

This surface interpolation algorithm is based on the solution to the hyperbolic multiquadric equations described in Smith [3]. The surface is known at the set of points x_i, y_i, z_i where $i \in \bar{H}$, and can be estimated at other points $h \in H$ by the formula

$$z(x_h, y_h) = \sum_{i \in \bar{H}} c_i * g(x_h - x_i, y_h - y_i),$$

where g is the basis function for the surface representation, and coefficients c_i are the solutions to the set of linear equations:

$$z(x_j, y_j) = \sum_{i \in \bar{H}} c_i * g(x_i - x_j, y_i - y_j) \text{ for all } j \in \bar{H}$$

Clearly, this irregular grid solution could be used to compute the surface values at the holes in the disparity data, but this involves solving for the coefficients c_j for each different configuration of holes and nonholes in the $n \times n$ neighborhoods of the disparity surface.

An alternative approach, which is used here, is to convert the quasi-regular grid problem into a regular grid problem in which each c_i at a hole is forced to be zero, and the corresponding z_i remains as an unknown. This results in the same solution that would have been obtained from the irregular grid formulation and produces the following system of linear equations:

$$\sum_{i \in H} A_{h,i}^{-1} * z_i = - \sum_{j \in \bar{H}} A_{h,j}^{-1} * z_j \text{ for all } h \in H, \quad (1)$$

where A^{-1} is the inverse of the matrix $A_{i,j} = g(x_i - x_j, y_i - y_j)$ for $i, j \in H \cup \bar{H}$. This system of equations must be solved for each z_i for $i \in H$. Thus, we have reduced the size of the linear system of equations that must be solved from the number of elements in \bar{H} to the number of elements in H . Of course, the matrix A must be computed and inverted once.

Areas on the disparity surface that contain large clusters of holes cause problems. The previous surface interpolation algorithm degenerates to a surface extrapolation algorithm when the nonholes in the neighborhood are not more or less isotropically distributed over the entire neighborhood. The problem can be overcome by increasing the size of the neighborhood until some spatial-distribution criterion is met, but this would require solving extremely large linear systems.

Large holes are filled by means of the following hierarchical approach:

Procedure Surface-Interpolate(surface_{*i*})

1. If surface_{*i*} contains large holes then

- (a) Compute filled-surface_{i+1} = *expand*(surface-interpolate(*reduce*(surface_i))), where *reduce* computes a Gaussian convolution reduction by a factor of two, surface-interpolate is a recursion call to this interpolation algorithm, and *expand* computes expansion by a factor of two, using bilinear interpolation.
 - (b) For each hole in surface_i that is completely surrounded by other holes, fill the hole with the value from the filled-surface_{i+1}.
2. For each hole in surface_i, fill the hole by solving the system of linear equations (1) for the $n \times n$ pixel neighborhood centered at the hole ($n = 7$ in the examples).
 3. Return the filled surface_i.

6 Examples

This section describes the experimental results achieved when the HWS technique was applied to areas of the ETL Phoenix Mountain Park data set, and compares these results to those obtained from the semiautomatic system developed by Norvelle [1].

The following components of the Phoenix Mountain Park data set were used:

- Left image: 2048 x 2048 pixels, 8 bits per pixel
- Right image: 2048 x 2048 pixels, 8 bits per pixel
- x-correspondence array: 400 x 400 points, floating point.

The left and right images had been scanned such that the epipolar lines were almost exactly horizontal. The ETL x-correspondence array was converted to an x-disparity image to enable comparison between ETL and HWS results.

Results are shown for two different areas of the Phoenix data set. All disparity measurements are indicated in terms of pixel distances in the 2048 x 2048 Phoenix stereo pair, rather than the resolution of the selected windows.

- Area A is defined by two approximately aligned 150 x 150-pixel windows of the Phoenix pairs which were reduced by a factor of four (the windows thus corresponding to the 600 x 600-pixel windows of the originals). The measured disparities for area A range from -40 to +16 pixels.
- Area B is defined by two approximately aligned 125 x 125-pixel windows of the Phoenix pairs which were reduced by a factor of two (the windows thus corresponding to the 250 x 250-pixel windows of the originals). The measured disparities for area B range from -40 to -34 pixels.

Figures 2 and 3 show the inputs and outputs of three levels of the hierarchy for areas A and B, respectively. Columns 1 and 2 are the reference and target images at each level. Column 3 is a binary image that indicates the positions of match failures. Column 4 shows the resulting disparity image of each level after the match failures have been replaced by surface-interpolated disparity values.

Figures 4 and 5 contain a comparison of the HWS results with those obtained at ETL by Norvelle for areas A and B respectively.

The bottom-left images of figures 4 and 5 show the pixel-by-pixel differences, after contrast enhancement, between the HWS and ETL disparities. The graphs to the right of these difference images depict the histograms of these differences.

The mean and standard-deviation values shown with the histograms provide a useful quantitative comparison between the HWS and ETL results. They show that the average disparity differences were .082 and .025 pixels, and that the standard deviations of the disparity differences were .67 and .34 pixels for the A and B window pairs, respectively, in terms of pixel distances in the 2048 x 2048 Phoenix pairs. These standard deviations become .17 and .17 pixels when expressed relative to the scales of A and B windows, respectively.

Similar results have been achieved for other examples that include both higher resolution and larger windows.

7 Problems

HWS is still very experimental. Some of the parameters that affect the system, such as the range of disparities to compute at each level of hierarchy and the size of the correlation operator, are still specified manually.

There are problems in estimating the range of disparities to be computed at each level of the hierarchy. If the estimate is too low, there will be frequent out-of-range match failures. If, on the other hand, the estimate is too high, computation time will increase and there will be more potential for match point ambiguity.

HWS has difficulty dealing with steep terrain features that have small image projections, but large disparities. At low resolutions in the matching hierarchy, the disparities of the terrain surrounding the feature dominate those of the feature itself, resulting in a disparity estimate that is usually intermediate between that of the feature and that of the surround. At higher resolutions in matching, the disparity of the steep feature may be outside the permissible disparity range.

HWS has even greater problems with oblique stereo pairs containing many occlusions. At low matching resolution, the disparities of foreground and background in the same neighborhoods cannot be distinguished. As the matching resolution increases, foreground and background features are discernible as separate objects, but their disparities are out of range for the matcher.

Most of the difficulties caused by sudden changes in disparity might be solved by preceding the disparity surface interpolation step with an algorithm that attempts to match still unmatched regions in the reference image with regions in the target image that likewise have not yet been matched. We thus attempt to match holes with holes.

8 Conclusions

HWS produces very good results for vertical stereo pairs of rolling terrain. With the inclusion of a hole-to-hole matching step, HWS should be capable of comparable performance for terrain characterized by steep slopes and frequent occlusions.

Bibliography

- [1] Norvelle, F. Raye, *Interactive Digital Correlation Techniques For Automatic Compilation of Elevation Data*, U.S. Army Engineer Topographic Laboratories, Fort Belvoir, VA 22060, Report Number ETL-0272, Oct. 1981.
- [2] Burt, Peter J., *Fast Filter Transforms for Image Processing*, CGIP, 16, 20-51. 1981.
- [3] Smith, Grahame .B., *A Fast Surface Interpolation Technique*, Technical Note 333, Artificial Intelligence Center, SRI International, Menlo Park, California, August 1984. (Also in these proceedings).

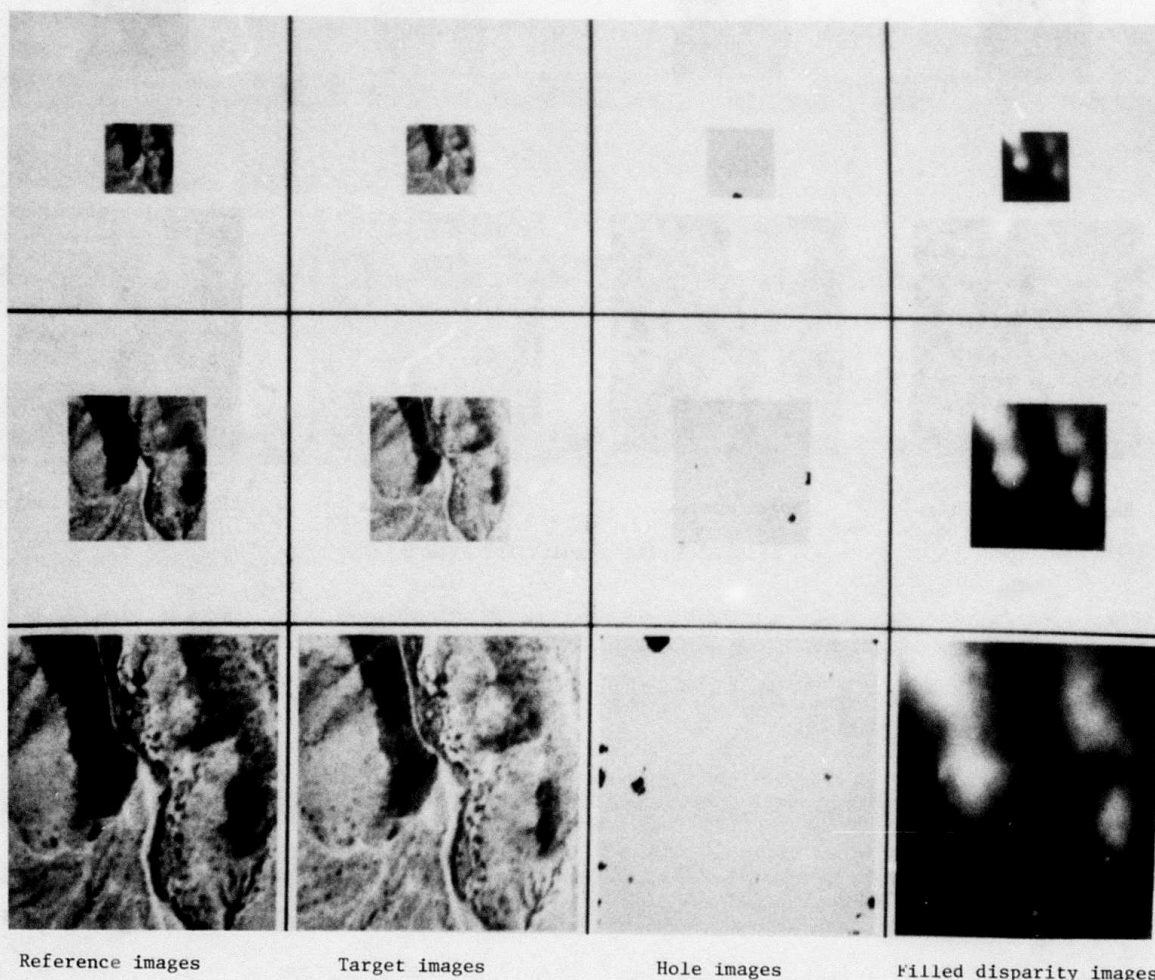


FIGURE 2 HWS results for area A

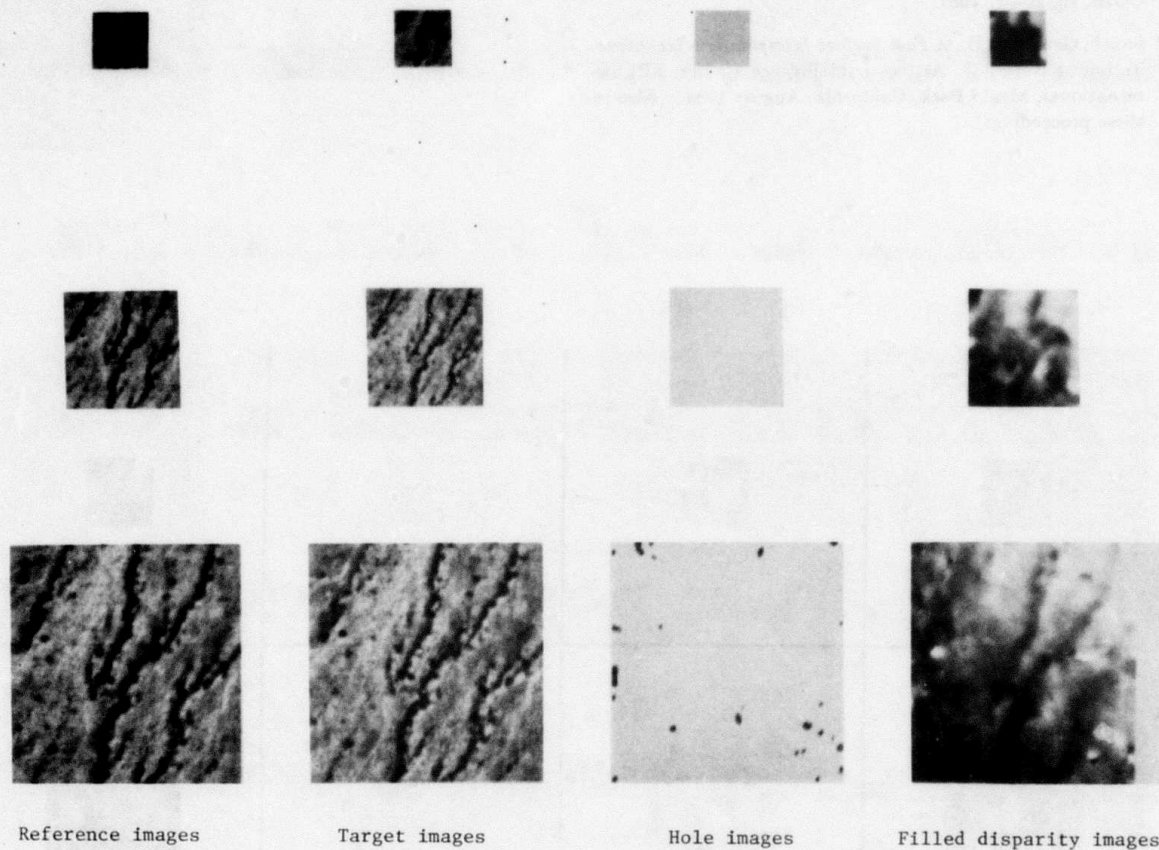
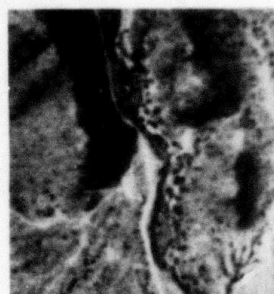


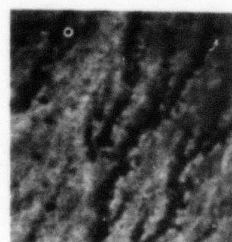
FIGURE 3 HWS results for area B



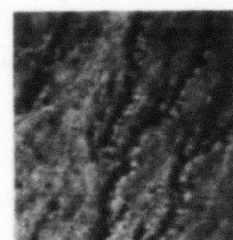
Reference image



Target image



Reference image



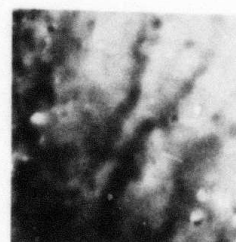
Target image



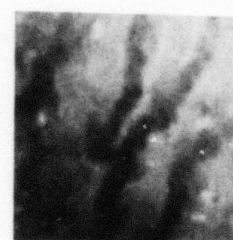
HWS disparity image



ETL disparity image



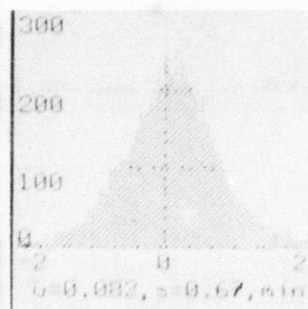
HWS disparity image



ETL disparity image

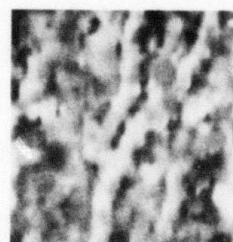


HWS - ETL difference

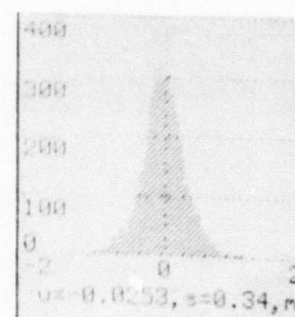


HWS - ETL histogram

FIGURE 4 HWS vs. ETL results for area A



HWS - ETL difference



HWS - ETL histogram

FIGURE 5 HWS vs. ETL results for area B

SECTION III

TECHNICAL PAPERS

NOT

PRESENTED

ON DETECTING EDGES

Vishvjit S. Nalwa

Artificial Intelligence Laboratory
Stanford University, CA 94305

Abstract

First, an edge is defined in terms of edgels i.e. short, linear segments, each characterized by a direction and a position. These, in turn, correspond to discontinuities in the intensity profile of the imaged scene. The problems associated with zero-crossings of the second derivative are then discussed. An alternate approach, which is a variant of surface-fitting, is presented. This fits a series of one-dimensional surfaces to each window, and accepts the surface with the smallest adequate basis. The tanh is an adequate basis for a step-edge, and its combinations are adequate for the roof-edge and the line-edge. This method is robust w.r.t. noise; it has subpixel position resolution and a 5° angle resolution; it is insensitive to high intensity gradients; and further, it takes into account convolution with a gaussian, which to a first approximation describes the imaging process. These claims are substantiated with processed images.

I. Introduction

It is hard to over-emphasize the importance of edge-detection in image understanding and yet, the common view in the community seems to be that the problem is far from solved.

We begin, in section II, by giving a definition of an edge in terms of the intensity profile which would be recorded by a perfect imaging system. Then, in section III, some of the problems associated with detection based on zero-crossings of the second derivative are discussed. Much of the work to date, has used a variant of this criteria. In this paper a variant of the surface-fitting approach is used; however, there are at least five significant differences from most previous approaches. 1) A *one-dimensional* surface, i.e. a surface constrained to be constant in one dimension, is used. It results in effectively filtering out the noise without affecting the edges. 2) It is not sought to mark pixels as belonging to an edge, but to detect short, linear pieces of edges, called edgels. Thus, to display our edges, which have sub-pixel localization, we use a finer grid than that of the image. 3) The blurring function of the imaging system, which is a gaussian to a first-order approximation, is

taken into account. This is considered more appropriate than edge-detection preceded by deconvolution, keeping in view the ill-conditioned nature of the latter. 4) An adequate basis has been found not only for most step-edges, but also for roof-edges and line-edges. These are various configurations of the tanh function. 5) Although it does not consider the problem of multiple detection for the first time, the philosophy is very different. We believe that if an edge-detector has sub-pixel localization, the multiple responses of each edge would lie within sub-pixel dimensions and any of them is a good enough estimate of the edge. This is, by all means, preferable to a single estimate which has worse localization. In fact, it may be possible to combine this wealth of information to get a refined estimate. These and other details are discussed in section IV. The precise algorithm, which has been implemented for step-edges, is listed step-wise in section V.

This approach to edge-detection is robust w.r.t. noise. For $(\text{step-size} / \sigma_{\text{noise}}) \geq 2$ it has subpixel position resolution and a 5° angle resolution. Further, it is insensitive to high intensity gradients which do not correspond to edges. These claims are substantiated with examples in section VI and we conclude with section VII.

It should be pointed out that the problems of linking and scale are not considered. It appears that these are not as involved as they may seem, if the the edgel-detection produces few false positives and false negatives.

II. Definition of an Edge

Any edge in an image can be thought of as short linear segments, called edgels, each characterized by a position and an angle. These edgels would correspond to local discontinuities of varying order in an image viewed with a perfect imaging system. To make ourselves clear, a discontinuity of the n^{th} order is one whose n^{th} derivative contains a delta function. Hence, the line-edge would contain



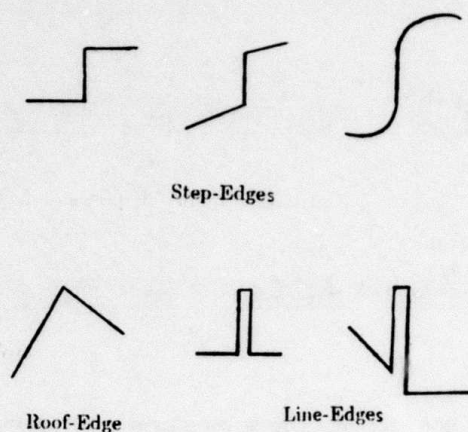


Fig. 1. Examples of Edges.

a discontinuity of the 0th order, the step-edge of the 1st order and the roof-edge of the 2nd order. Some examples of these are shown in Fig. 1. However, the images we obtain in practice are far from perfect. Not only are they noisy, but also degraded due to optical and other aberrations. These effects can be represented to a first-order approximation, by convolution with a gaussian [Andr], having a certain standard deviation. This is fortunate because it bandlimits the signal before sampling it. Absence of this *blur* would result in severe aliasing. However, this affects the edges too. There are no discontinuities in the image anymore. This is of consequence, as will be noted in the section on the zero-crossings of the second derivative.

III. Zero-Crossings of the Second Derivative

Much of the work to date has used zero-crossings of the second derivative to detect and/or localize edges [Cann, Hara etc.]. The problems associated with this are many. Any derivative is inherently a noise sensitive operator. If the window fitting technique is used, and the basis is inadequate, as will often be the case, the zero-crossing will result in extremely bad localization. For example, the typical error for a cubic basis used to detect a step edge is a few pixels.

Further, it's not hard to see that we can have zero-crossings in the absence of an edge e.g. at the base of a ramp [Binf]. Zero-crossings of the second-derivative are basically points of inflexion and these need not correspond to edges as in the case of a corrugated intensity surface.

It will be a rare occurrence for a step-edge to correspond to an underlying mathematical step. We would expect that the intensity surface on the two sides of the step would in general be sloped as shown in Fig. 1, even if

the slope is slight. A simple analysis (Appendix I) of this generalized step convolved with a gaussian, shows, that in cases where both sides of the step don't have the same slope, the localization based on zero-crossings would be biased by $(\Delta_{slope} \cdot \sigma_{gaussian-blur}^2 / step-size)$. On more than one occasion, authors have suggested convolving the image with a gaussian before finding the zero-crossings. It can be shown that this would effectively amount to having a blurring function with a variance equal to the sum of the two variances and hence degrade localization.

IV. The Details

A variant of the surface-fitting approach is used here. This fits a surface to the intensity-levels of the pixels and was popularized by Haralick [Hara]. However, he has used a cubic surface, which is an inadequate basis for most edges and has a typical localization error of a few pixels. It further presents the problem of multiple detection.

Recently, Canny [Cann] has applied an interesting functional approach to edge-detection. The criteria are inherently justifiable and yet there is no reason why one should use a linear filter, to meet them. Our view on the problem of multiple detection has been stated in section I. We believe that sub-pixel localization makes the problem redundant. Further, most authors have ignored the fact that the image consists of samples of the true intensity profile blurred by the imaging system. The standard deviation of this gaussian blurring function can be determined by examination of the image of a point-source or step-edge. As a result of this blur, we have an image with no underlying discontinuities. The spectrum is bandlimited and aliasing avoided.

The noise is generally assumed to be additive gaussian. Some authors have tried to reduce this, by convolving the image with a gaussian. As discussed in the previous section, this deteriorates the localization. If one could find the direction of the edge in a reliable fashion, then the noise could be reduced by smoothing the window in a direction parallel to the edge. This, of course relies on the fact that the window is small enough to justify the presence of edgels. We have achieved this smoothing, by fitting to each window a *one-dimensional* surface i.e. a surface which varies only in one direction as shown in Fig. 2. This direction would be perpendicular to the that of the edgel in the window, if any.

So, now we come to the question of a reliable direction-finder, for windows hypothesized to contain edgels. A first-approximation for the direction can be obtained by fitting a planar surface to the window. This was found to have $\sigma_{direction} < 10^\circ$ for edges with $(step-size / \sigma_{noise}) = 2$. A more general method can be used to refine this first estimate. We fit a *one-dimensional* cubic surface to the nearest 5° and choose the one with the minimum square-error (the

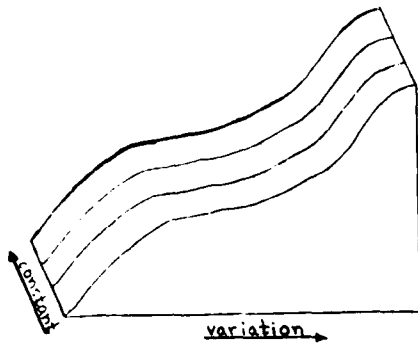


Fig. 2. One-Dimensional Surface

corresponding equation is stated in the appendix). Starting with the initial estimate, this search is generally not more than a few steps. It is important to point out that the plot of the square-error vs angle is bowl-shaped and centered around the true angle, for windows containing edgels. Hence, once within the bowl, the standard methods like steepest-descent can be used to find the minima. For edges with the step size twice the standard deviation of the noise the variance is approximately 5° .

It should be pointed out that there cannot be any one unique basis which can be used for all the windows. If we attempt to do this, we will obtain incorrect results when the basis is inadequate and noise sensitive results if the basis is not minimal. Similar considerations for one dimensional steps have been investigated by Leclerc [Lecl].

Now, the choice of an adequate basis. For most step-edges the tanh function will be adequate. One important by-product of employing the tanh is the contrast of the edge. From our case studies it seems that this would be helpful not only in linking, but also interpretation. However, for edges which do not have similar slopes on both sides of the step, the tanh is inadequate and a cubic or a tanh with a cubic might be adequate. The latter has some problems because the tanh and cubic are not completely independent. It may be desirable to employ splines when the tanh and the cubic are inadequate bases. It should be repeated, that the cubic is inadequate for most step edges and that the derivative is not a very noise-robust operator. As a result even if one does use a cubic, it is preferable to localize and obtain the contrast from the tanh fit. We have used a cubic for our detector because our window is too small (5×5), for finding the parameters of a tanh with a cubic or of splines, in the case of horizontal and vertical edges. For roof-edges and line-edges, combinations of the tanh function, as depicted in Fig. 3, seem to be adequate. We compare the quadratic fit with the tanh fit to determine the existence of a step-edgel. In the initial stages, we had used the Chi-Square statistics to determine the adequacy of the basis. It was found that this was unnecessary and perhaps undesirable because edges of high contrast seemed to be more noisy.

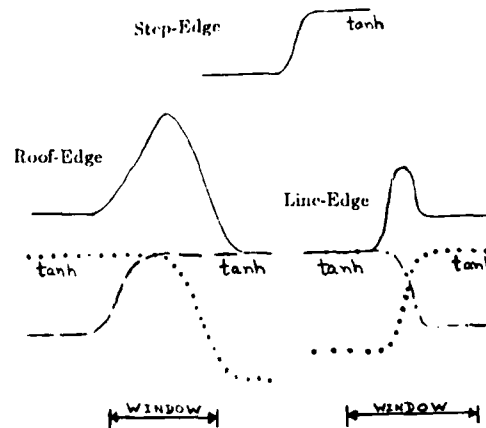


Fig. 3. Adequate Bases

Some authors have tried to compensate for an inaccurate model, i.e. a mathematical step for a step-edge, by using different scales. Different scales, we believe, are unnecessary for most scenes completely within the depth-of-field of the camera, unless some of the shadows are particularly diffuse.

We have carried out our initial investigation for step-edges, with images having a single scale i.e. all objects are within the depth-of-field of the camera. The blurring function for our camera was determined to have a standard deviation of 0.6 pixels. Numerically, it was determined that for a standard deviation of 1 and a mathematical step, the optimum scaling factor for the argument of the tanh function was 0.86. This factor was determined by minimizing the square-error. Hence, a rule of the thumb for the scaling factor is $(0.86 / \sigma_{\text{gaussian blur}})$. The detection scheme is not particularly sensitive to this factor and in fact it detects reasonably diffuse shadows.

The window size is determined by the standard deviation of the blurring gaussian. As the window size is increased for a fixed blur, we trade-off resolution for localization. Resolution refers to the minimum support required for the detection of an edgel i.e. if an edgel can be located within any window without the simultaneous presence of any other edgel, it is theoretically resolvable. If it is not detected, it would be due to the inadequacy of the edge-detector. If three parallel edges are two pixels apart, then the middle edge would not be resolvable, but the other two might be. We will point out examples of these in our first case study.

V. The Algorithm for Step-Edgel Detection

- (i) Fit a planar surface to the window, minimizing the

square-error, in order to get an estimate of the edgel's directionality, if one exists.

- (ii) Refine the estimate, using a one-dimensional cubic surface with the same error criteria. The resulting equations are non-linear in the angle. However, owing to the reliable initial estimate, this is typically a couple of search steps. We find the angle to the nearest 5° . Calculate the 2, 20 F-Statistic for the planar and cubic fits. If it exceeds the 75% threshold, then declare the absence of an edgel.
- (iii) Fit an optimal tanh 1-D surface in the direction found in (ii). The tanh is localized to the nearest 0.1 pixel.
- (iv) Fit a quadratic over the window and compare with a tanh-fit. If the square-error in this case is less i.e. the 21, 21 F-Statistic is less than the 50% threshold, then declare the absence of an edgel.
- (v) The step-size is determined by the coefficient of the tanh-fit and the position, by its origin. The intensity values on the two sides of the step can be found from the constant in the basis.
- (vi) Repeat for all the pixels, by shifting the window in steps of one in the x and y directions.

N.B. All the relevant equations and statistics are listed in the Appendix II. If one wants to detect step-edges which have significantly different slopes on the two sides of the edgel, steps similar to (iii) and (iv), but with a basis different than the tanh, will have to be added.

VI. Three Case-Studies

The proof of the pudding is in the eating. It is essential to point out some details concerning the photographs displayed. a) Only the step-edgel detector has been implemented. b) The pictures with the edges are displayed on twice the frame size of the pictures with only the original image, for reasons mentioned in the introduction. c) These pictures have the intensity of the edges proportional to their contrast. d) The edges displayed in all cases have been thresholded, for contrasts less than twice the standard deviation of the noise. e) All edges displayed, are composed of the raw edges with no post-processing, like linking, thinning, cleaning etc.. f) When comparing the performance of our approach to that of its predecessors, it would be desirable to note the size of the original image. g) The images with the edges superposed, are formed by enlarging the original image to four times its area and by making the edge pixels have the lowest gray level. It is instructive to refer to the original image, the edgel image and the superposed image simultaneously to scrutinize the performance of the edge detector. h) While doing so, it is important to bear in mind that the declaration of false pos-

itives and false negatives should be based on the original image and should not be influenced by our preconceived notions of regular and suggestive shapes.

(i) Industrial Setting : Bin of Parts

Size of the Original Image : 128 x 128
Standard Deviation of the Blur : 0.6
Standard Deviation of the Noise : 5

Fig. 4-a : The Original Image

Fig. 4-b : The Edgel-Image

The pins of the various parts have false negatives. This is because they are bounded by dark lines, and our edge detector has currently been implemented only for step-edges. The outer edge of the lines have been detected, but the inner edge exceeds the resolution capabilities of our detector. Also, notice that some of the circular regions detected have a diameter of just a few pixels.

Fig. 4-c : The Superposed Image
Notice the localization.

(ii) Aerial View : San Francisco Bay

Size of the Original Image : 256 x 256
Standard Deviation of the Blur : 0.6
Standard Deviation of the Noise : 5

Fig. 5-a : The Original Image

This picture was chosen because of its complexity.

Fig. 5-b : The Edgel Image

On a first glance, it may seem to a casual observer that there are a host of false positives. However, on a closer examination this seems to be untrue. The long lines in the sea correspond to silt lines. It probably will not be possible to confirm these in the photographs you will see. Note, the continuity in most edges. It is unlikely to have long continuous false positives. Further notice the bridge in the upper left corner.

Fig. 5-c : The Superposed Image

Now, note the localization and the structure imposed by the edges. Compare the intensity on the two sides of each edgel. Use the information provided by all three images, once again, while scrutinizing the performance.

Indoor Scene : A Telephone, a Cup and a Pencil.

Size of the Original Image : 256 x 256
Standard Deviation of the Blur : 0.6
Standard Deviation of the Noise : 4

Fig. 6-a : The Original Image

This image was chosen to illustrate our case against the misuse of scale. Note the top surface of the telephone. It does not correspond to a mathematical step, but to a generalized step, which most detectors would find by using different scales. Also note, the top edge of the book on the left of the table and the top edge of the pencil.

Fig. 6-b : The Edgel Image using Tanh-fit

Note the missing edges, which were pointed out in the preceding paragraph. These can be obtained using a larger scale, but we feel that this is not only avoidable, but also unnecessary. The inner portion of the flower on the cup has a few false negatives due to lack of resolution.

Fig. 6-c : The Edgel Image using Tanh / Cubic

Notice the detection of the missing edges. However, now we have multiple detection, i.e. the edges are thicker than they would be without the cubic, at some places.

Fig. 6-d : The Superposed Image (using Fig. 6-c)

Notice the localization, once again.

VII. Conclusion

This paper attempted the problem of edge-detection using *one-dimensional* surfaces. Edges were defined in terms of short, linear segments, called edgels. Detection of edgels seems to be more appropriate than of edge-pixels because edgels not only directly capture the characteristics of edges, but they also establish continuity. An adequate basis for most step-edgels was claimed to be the tanh. Robustness to noise, sub-pixel position localization and 5° angle localization were also claimed. A case against derivative operators and the unnecessary use of multiple scales was presented. Some case studies, with the algorithm for step-edgels implemented for a single scale, were then examined.

Before we conclude, we would like to make two interesting observations. The blur, caused by optical aberrations and the scanning mechanism, is, to a certain extent, needed to avoid *undersampling*. In the C.C.D. cameras, sharp focussing produces aliasing. This is because, the image in these cameras, is *scanned* by rectangular non-overlapping windows. In such situations, slight defocussing i.e. just enough to avoid aliasing, is desirable!

Edges in different orientations are not equal. This is because of the use of a rectangular grid. Edges at different angles, have varying localization capabilities. The reason is not hard to see. Edges, at angles close to 0° and 90° , have samples at a minimal number of points across the edge. The edges at 45° also exhibit coincidental alignment, however this is much less than the former. Another

way to look at it is, that if we were trying to determine the parameters of an adequate basis, edgels at 0° and 90° would require the largest minimal windows and edgels at 45° would come next. It suggests, that for everyday scenes, with most edges in the horizontal and vertical directions, it is advisable to tilt the camera!

As noted by Binford [Binf], current edge-detection schemes have primitive capabilities, at best, and *nature has enough variety to find out a kludge and make us pay for it*. This paper would consider its purpose served, if it has succeeded in highlighting some of the issues and concerns in edge-detection. It does not serve any purpose, at this juncture, to talk about the computational requirements of this algorithm, not only because it is yet to be considered from an efficiency point of view, but more so because we will finally have to resort to parallel and more efficient architectures. This algorithm has a natural extension for roof and line edges, which is yet to be implemented. Work also needs to be done on an adequate basis for step-edgels which have large deviations from mathematical steps.

Acknowledgement

Support for this research was provided by ARPA, under contract N00039-84-C-0211, and by the Information Systems Laboratory at Stanford. The author would like to express his gratitude to Prof. Thomas O. Binford, not only for numerous thought-provoking discussions and sharing his considerable insight and experience with this problem, but more so for leading him to perceive the sharp distinction between scientific investigation and speculation. Time and again, assistance was received from Jitendra Malik and Sathyanarayanan. If it was not for the constant encouragement of Ron Fearing, this paper might not have happened.

Appendix I : Zero-Crossing Bias

Let, the step-edge be $E(x)$ and the Gaussian be $G(x)$.

$$E(x) = \begin{cases} k_1 \cdot x & \text{if } x < 0 \\ k_2 \cdot x + s & \text{if } x > 0 \end{cases}$$

$$G(x) = e^{-x^2/2\sigma^2}$$

It can be shown, that $(E(x) * G(x))'' = E(x) * G''(x)$.

$$\begin{aligned} E(x) * G''(x) &= \int_{-\infty}^{\infty} k_1(x-u) \cdot G''(u) du \\ &\quad + \int_{-\infty}^{\infty} [k_2(x-u) + s] \cdot G''(u) du \\ &= \int_{-\infty}^{+\infty} k_1(x-u) \cdot G''(u) du \\ &\quad + \int_{-\infty}^{\infty} [(k_2 - k_1) \cdot (x-u) + s] \cdot G''(u) du \end{aligned}$$

$$\begin{aligned}
&= [s - (k_2 - k_1) \cdot x] \cdot G'(x) \\
&\quad + (k_2 - k_1) \cdot [x \cdot G'(x) - G(x)] \\
&= s \cdot G'(x) - (k_2 - k_1) \cdot G(x)
\end{aligned}$$

Equating this to zero we get, $x = \Delta_{slope} \cdot \sigma^2 / s$ where $\Delta_{slope} = (k_1 - k_2)$ and s is the step-size. This is the biased zero-crossing of the second derivative.

Appendix II : Least-Squares Criteria and Statistics

Least-Squares Criterion for a Planar-fit :

$$EtaP = \sum_{x,y=0}^4 (Image[x,y] - (a_0 + a_x \cdot x + a_y \cdot y))^2$$

(minimize w.r.t. a_0, a_x and a_y)

Initial Estimate of b : $b_0 = \arctan(a_y/a_x)$

Least-Squares Criterion for a Quadratic-fit :

$$EtaQ = \sum_{x,y=0}^4 (Image[x,y] - (a_0 + a_1 \cdot z + a_2 \cdot z^2))^2$$

(minimize w.r.t. a_0, a_1 and a_2)

$$z = x \cdot \cos(b) + y \cdot \sin(b)$$

b is determined from the L.S.E. cubic-fit and is the angle by which the axes have to be rotated to align the x-axis with the edgel cross-section.

Least-Squares Criterion for a Cubic-fit :

$$EtaC = \sum_{x,y=0}^4 (Image[x,y] - (a_0 + a_1 \cdot z + a_2 \cdot z^2 + a_3 \cdot z^3))^2$$

(minimize w.r.t. a_0, a_1, a_2 and a_3)

$$z = x \cdot \cos(b) + y \cdot \sin(b)$$

The initial estimate of b , from the L.S.E. planar-fit, is refined. The equations to be solved are non-linear in b

Least-Squares Criterion for a Tanh-fit :

$$EtaT = \sum_{x,y=0}^4 (Image[x,y] - (s \cdot \tanh(f \cdot [z+p]) + k))^2$$

(minimize w.r.t. s, p and k)

f is determined from the rule of thumb mentioned in section IV i.e. $(0.86 / \sigma_{\text{gaussian-blur}})$. Twice s is the edge contrast and p is the position.

Statistics :

EtaP follows the Chi-Square Statistic with 22 D.O.F..

EtaQ follows the Chi-Square Statistic with approximately 21 D.O.F..

EtaC follows the Chi-Square Statistic with 20 D.O.F..

EtaT follows the Chi-Square Statistic with approximately 21 D.O.F..

$\{10 \cdot (EtaP - EtaC) / EtaC\}$ follows the 2, 21 F-Statistic.

$\{EtaQ / EtaT\}$ follows the 21, 21 F-Statistic.

References

- Andr : Andrews, H.C. and Hunt, B.R.: "Digital Image Restoration," Prentice-Hall Inc., Englewood Cliffs, 1977.
- Binf : Binford, T.O.: "Inferring Surfaces from Images," Artificial Intelligence Journal, August 1981.
- Cann : Canny, F.J.: "Finding Edges and Lines in Images" AI-TR 720, M.I.T. A.I. Lab., June 1983.
- Hara : Haralick, R.M.: "Digital Step Edges from Zero-Crossings of Second Directional Derivatives," IEEE Trans. P.A.M.I.-6, No. 1, Jan. 1984.
- Lecl : Leclerc, Y. and Zucker, S.W.: "The Local Structure of Image Discontinuities in One Dimension," TR-83-19R, Computer Vision and Robotics Lab., McGill Univ., May 1984.

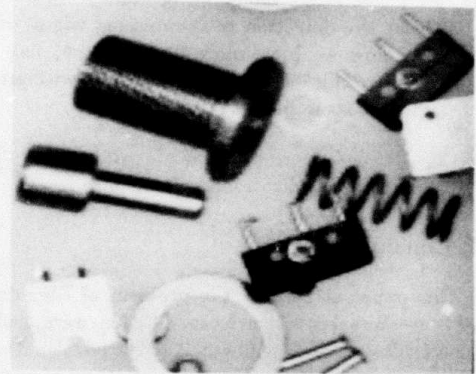


Fig. 4-a. Bin of Parts : Original-Image



Fig. 4-b. Bin of Parts : Edgel-Image

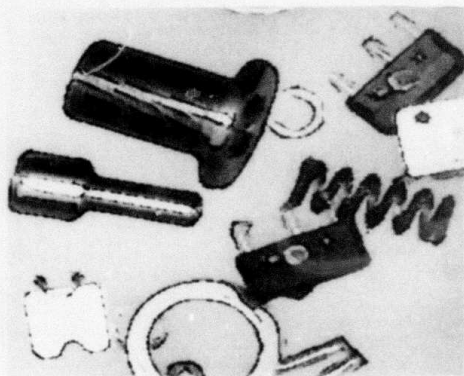


Fig. 4-c. Bin of Parts : Superposed-Image

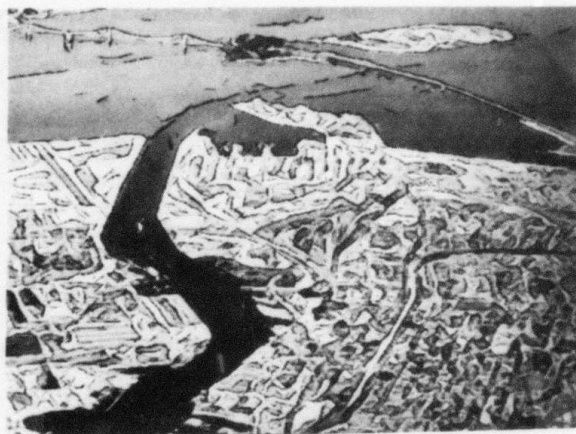


Fig. 5-c. San Francisco Bay : Superposed-Image



Fig. 5-a. San Francisco Bay : Original-Image

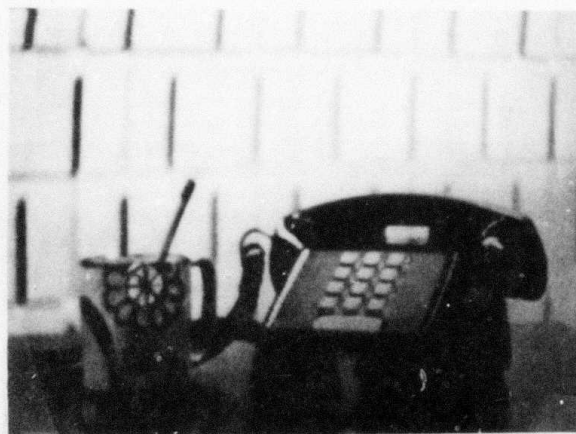


Fig. 6-a. Indoor Scene : Original-Image



Fig. 5-b. San Francisco Bay : Edgel-Image

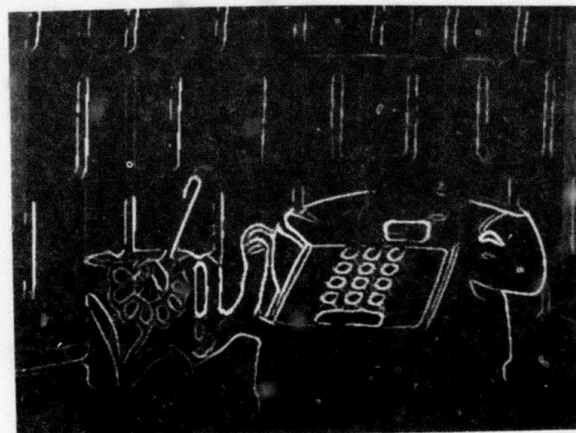


Fig. 6-b. Indoor Scene : Edgel-Image (tanh)

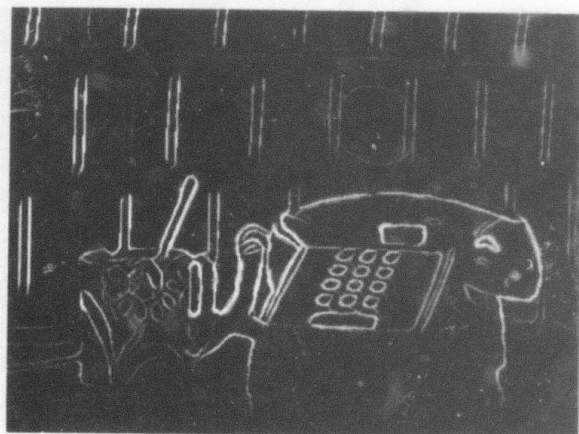


Fig. 6-c. Indoor Scene : Edgel-Image (\tanh/\cubic)

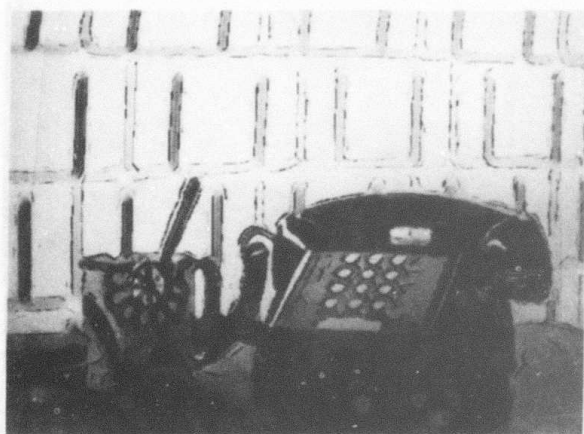


Fig. 6-d. Indoor Scene : Superposed-Image

EXTRACTING STRAIGHT LINES

J. Brian Burns, Allen R. Hanson, Edward M. Riseman

Computer and Information Science Department
University of Massachusetts
Amherst, Massachusetts 01003

ABSTRACT

This paper presents a new approach to the extraction of straight lines in intensity images. Pixels are grouped into edge support regions of similar gradient orientation, and then the structure of the associated intensity surface is used to determine the location and properties of the edge. The resulting regions and extracted parameters form two separate representations of a straight line segment, pixel-based and symbolic, that can be used together for a variety of purposes. The algorithm appears to be far more effective than previous techniques for two key reasons: 1) the gradient orientation is used as the initial organizing criterion in the extraction of straight lines, instead of the gradient magnitude; and 2) data directed organization of the complete context of a straight line is determined prior to any local decisions about participating edge points.

1.0 INTRODUCTION

The organization of significant local intensity changes into the more global abstractions called "lines" or "global intensity boundaries" is an early, but important, step in the transformation of the visual signal into useful intermediate constructs for interpretation processes. Despite the large amount of research appearing in the literature, effective extraction of linear boundaries has remained a difficult problem in many image domains. There are two goals of this paper: a) the development of mechanisms for extracting straight lines from complex images, including intensity discontinuities of arbitrarily low contrast; and b) the construction of an intermediate representation of edge/line information which allows high-level mechanisms efficient access to relevant lines. A more detailed presentation can be found in [BUR84].

We contend that the major failings of line extraction algorithms are twofold: the relegation of information about edge orientation to a secondary role in the processing, and the lack of a comprehensive global view of the underlying image structure prior to making local decisions about edge features.

In most edge and line extraction algorithms, the magnitude of the intensity change has been used in some manner as the dominant measure of importance of the local edge. It is our view that edge orientation carries important information about the spatial extent of the straight line.

The technique presented here was motivated by a need for a straight line extraction method which can find straight lines in reasonably complex images, particularly those lines that are long but not necessarily of high contrast. A key characteristic of the approach presented here that distinguishes it from most previous work is the global organization of the supporting edge context prior to any decisions about the relevance of local intensity changes. An estimate of the local gradient orientation at each pixel is the basis of these first organizing processes. Grouping pixels into edge support regions avoids the *plethora of magnitude responses* from masks at varying sizes and orientations, as well as unnecessary complexity in the subsequent organizing mechanisms. The approach presented here has its roots in the "gradient collection" process of Hanson et al [HAN80], as well as [R78].

Our approach allows the extraction of straight lines despite weaknesses in line clarity due to local edge inconsistencies or deficiencies in width and contrast. It directly addresses the problems associated with the size of the edge operators and determines the extent of support to be given to edges and lines directly from the underlying data.

2.0 A REPRESENTATION AND PROCESS FOR EXTRACTING STRAIGHT LINES

2.1 Overview

There are four basic steps in extracting straight lines:

1. Group pixels into edge-support regions based on similarity of gradient orientation. This allows data directed organization of edge contexts without commitment to masks of a particular size.
2. Approximate the intensity surface by a weighted planar fit. The fit is weighted by the gradient magnitude associated with the pixels so that intensities in the steepest part of the edge will dominate.

3. **Extract** attributes from the edge-support region and the plane fit. The attributes extracted include the representative line and its length, contrast, width, location, orientation, and straightness.
4. **Filter** on the attributes to isolate various image events such as long straight lines; high contrast short lines (heavy texture); low contrast short lines (light texture); and lines at particular orientations and positions.

2.2 Grouping Pixels into Edge Support Regions Via Gradient Orientation

Figure 1 shows two representative images used to illustrate the process. Figure 2(a) is a surface plot of a 32x32 subimage from another house image; results for the full images are shown in subsequent sections. The vector field drawn in figure 2(b) shows the corresponding gradient image where the length of the vector encodes gradient magnitude. The gradient estimates were formed by convolving the image with two-by-two edge masks (figure 2(b) inset). Note that the sign of the gradient is relevant.

An extremely simple process was employed to group the local gradients into regions on the basis of the orientation estimates. The 360 degree range of gradient directions is arbitrarily partitioned into a small set of regular intervals, say eight 45 degree partitions or sixteen 22.5 degree partitions. If our conjectures about edge orientation are correct, then pixels participating in the edge-support context of a straight line will tend to be in the same edge orientation partitions and adjacent pixels that are not part of a straight line will tend to have different orientations. A simple connected components algorithm can be used to form distinct region labels for groups of adjacent pixels with the same orientation label (Figure 2(c)). Note that in Figure 2(c) the great degree of fragmentation into many small regions of very low gradient magnitude could be grouped into a homogeneous region later, rather than interpreting them as edge elements.

To make the fixed partition technique more sensitive to edges of any orientation, the current algorithm uses two overlapping sets of partitions, with one set rotated a half-partition interval. Thus, if a 45 degree partition is used starting at 0 degrees, then a second set of 45 degree partitions starting at 22.5 is also used. The critical problem of this approach is merging the two representations in such a way that a single edge is principally associated with a single gradient region. The following scheme is used to select such regions for each edge: first the lengths are determined for the regions; then, since each pixel is a member of exactly two regions (one in each segmentation), the pixel decides which one provides the longest interpretation; finally each region counts up the number of pixels within its boundaries that voted for it as opposed to regions of the other segmentation. The 'support' a region is given is the number of votes for it over the total number of pixels in the region. The regions selected are those that have a majority, i.e., the support is greater than 50%. For further discussions on grouping see [BUR84].

2.3 Interpreting the Edge-Support Region as a Straight Line

The underlying intensity surface of each gradient region is a candidate for a straight line structure; the key problem is to use this information to find the line. The positional parameters extracted will serve as the core of the structure's symbolic description as well as a coordinate system about which other attributes will be measured.

In this section, we will examine a simple process for computing the parameters of a planar fit to the intensity surface of the pixels in each edge-support region. The region depicted in figure 3(a) and as dots in the surface plot of figure 2(a) will serve as our example. Note that it includes pixels outside the group of gradients depicted in figure 2(c), since the two-by-two masks incorporated them in the gradient estimation. Haralick [HAR81] modelled the local intensity surface in the neighborhood of a pixel as a planar surface patch called a 'sloped facet'. This planar fit served as a model of the region structure and was used to determine if the pixel was at a region boundary or not. In our application the planar fit will be applied to all pixels in a support region instead of an a priori fixed geometric configuration. If a direct least-square planar fit to all pixels in the support region is computed, then many pixels which might be at the tail of the intensity change could dominate the fit. Therefore, the pixels were weighted by local gradient magnitude to enhance the effect of points near the edge center.

An obvious constraint on the orientation of the line is that it be perpendicular to the gradient of the fitted plane. Thus, this leaves the problem of locating the line along the projection of the gradient. A simple approach is to intersect the fitted plane with a horizontal plane representing the average intensity of the region weighted by local gradient magnitude as shown in Figure 3(b); the straight line resulting from the intersection of the two planes is shown in Figure 3(a).

2.4 Extracting Attributes of the Support Context

The gradient region and the planar fit of the associated intensity surface provides the basic information necessary to quantify a variety of attributes beyond the basic orientation and position parameters. Length is simply the distance between the two endpoints. Other attributes of the line include properties of the intensity profile perpendicular to the line and its behavior along the length of the line. Analysis of the profile of the line can provide a measure of the edge's contrast and width (fuzziness), while behavior along the length determines its straightness; see [BUR84].

3.0 EXPERIMENTAL RESULTS

The algorithm described in the preceding sections was applied to the full images shown in Figure 1. The algorithm utilized overlapping partitions as described in Section 2.2; the partition size was 45 degrees, staggered by 22.5 degrees. Figures 4-5 demonstrate the performance of the algorithm.

Figure 4(a) shows the unfiltered output of the algorithm applied to the first house image. Note that all of the small and low contrast edges are still present. Figures 4(b-c) show the result of filtering 4(a) on the basis of gradient steepness (change in gray-levels per pixel) followed by a filtering on length that separates the edges into two disjoint sets, one corresponding to short texture edges (Figure 4b) and the other to longer lines related to the surface structure of objects in the image (Figure 4c). We are also examining ways in which texture descriptors may be constructed from the edge set remaining when a filter similar to that which produced 4(c) is applied to the initial edge data. In Figure 4c, the structural edges representing the telephone wires were extracted from a thin, one pixel wide diagonally oriented region, a difficult problem for many line extraction processes.

Figure 5 shows results on another typical house image, this time filtered on length alone (all edges greater than 5 pixels).

4.0 CONCLUSIONS

This paper has presented a low-level representation for straight lines. The technique for extracting straight lines is effective because it globally organizes the spatial extent of a straight line without local decisions about the meaningfulness of an edge feature. It does this by utilizing gradient orientation to provide a gradient segmentation of the pixels in the formation of edge-support regions. Analysis of the intensity surface of the pixels in these regions yields the information required to extract lines and characterize the intensity variations in a variety of ways. The algorithm is very robust and accurately extracts many low contrast long lines.

While the extracted lines, such as long straight lines, might be directly useful, the underlying edge-support region has a wealth of information useful to intermediate processing strategies. These include additional grouping mechanisms for linking co-linear straight line segments and linking piecewise-linear approximations to curved lines bounding areas of similar properties. It also contains information useful for grouping lines with common properties into textured regions. The representation can serve as a simple yet rich edge-line "primal sketch" [MAR77]. The edge-support regions might be useful in separating the straight lines into intrinsic images [BAR78] representing boundaries of different types such as illumination, texture, reflectance, orientation, etc.

5.0 REFERENCES

- [BAR78] Barrow, H.G. and Tenenbaum, J.M., "Recovering Intrinsic Scene Characteristics from Images," in Computer Vision Systems (A. Hanson and E. Riseman, eds.), Academic Press, 1978, pp. 3-26.
- [BUR84] Burns, J., Hanson, A. and Riseman, E., "Extracting Straight Lines," COINS Technical Report, University of Massachusetts, January 1984.
- [EHR78] Ehrlich, R.W. and Foith, J.P., "Topology and Semantics of Intensity Arrays," in Computer Vision Systems (A. Hanson and E. Riseman, eds.), Academic Press, 1978, pp. 111-127.
- [HAN80] Hanson, A., Riseman, E., and Glazer, F., "Edge Relaxation and Boundary Continuity," COINS Technical Report 80-11, University of Massachusetts, May 1980.
- [HAR81] Haralick, R.M. and Watson, L., "A Facet Model for Image Data," Computer Graphics and Image Processing, Volume 15, 1981, pp 113-129.
- [MAR77] Marr, D. and Nishihara, H.K., "Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes," Proc. Royal Society B, 200, 1977, pp. 269-294.

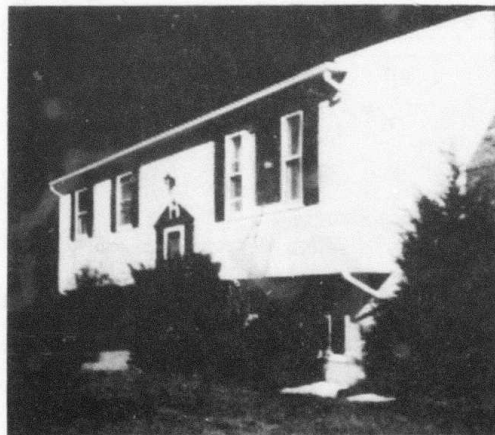
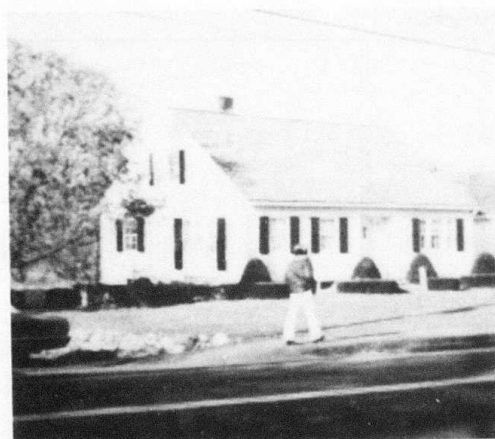


Figure 1. Two images used to demonstrate the straight line finder.

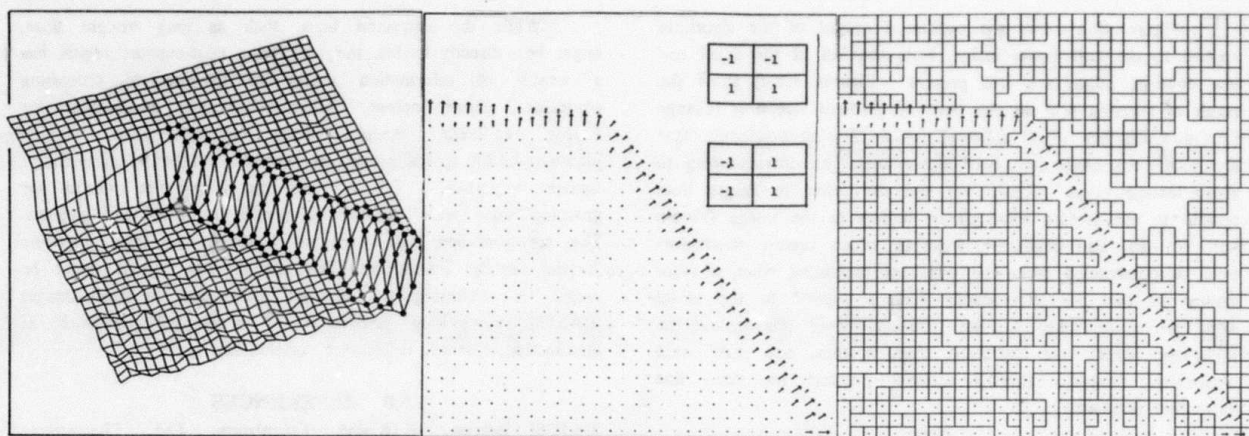


Figure 2. Forming gradient regions. (a) A 32x32 subarea of a house image which will be used to illustrate the process. (b) The 2x2 operators used to estimate dI/dy and dI/dx , from which the local gradient orientation is obtained and the resulting gradient vectors. (c) Gradient regions formed by a regular partitioning of the data into eight orientation classes.

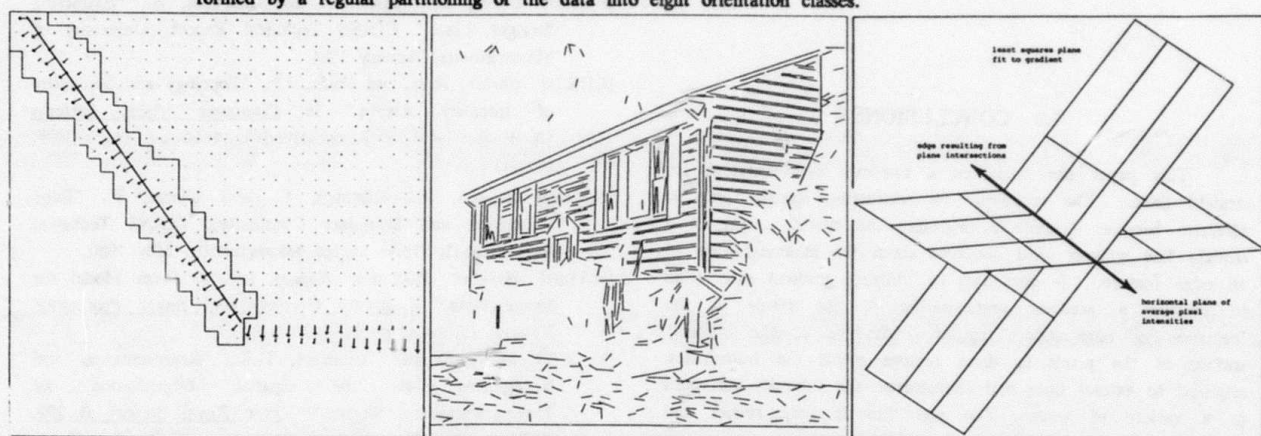


Figure 3. Obtaining straight lines. (a) A group of pixels associated with a gradient region from figure 2 and its interpreted straight line. (b) The straight line obtained by algorithm on the house image of Figure 1(b), intersecting the L.S.E. plane fit to the intensities, weighted by the gradient magnitudes, with the horizontal plane representing the average intensity, weighted again by magnitude.



Figure 4. Results of the line extraction algorithm on the house image of Figure 1(a). (a) Initial edge set. (b) Filtering (a) on gradient steepness (≥ 10 gray levels per pixel) and for shortness ($\text{length} \leq 5$). (c) Steep, long lines ($\text{length} \geq 15$ and gradient steepness ≥ 10 levels per pixel).

MATCHING CLOSED CONTOURS¹

Keith E. Price
Intelligent Systems Group
University of Southern California
Los Angeles, California 90089-0272

Index terms: Image analysis, Contour matching, Segment matching.

ABSTRACT

Many objects are recognizable by outlines of their two dimensional projections. If there are distinct features, the position and orientation can be determined, even when there are substantial occlusions or additional touching objects. Contour matching has been used for a variety of tasks by other researchers. We present a simple algorithm for matching linear representations of closed, or almost closed, boundaries of objects. Arbitrary changes in orientation and position are allowed along with occlusions. Unlike many other methods, no relaxation (or iterative updating of the match rating) is necessary. A complete system which uses multiple resolution representations (currently three) has been implemented and tested on a variety of scenes. The results for the general matching problem (determine if two contours can match and how to transform them to best match) are very good. Further work remains in using this method for identifying very similar objects, but for distinct objects it currently works very well.

1. INTRODUCTION

Two dimensional projections of objects are sufficient for many recognition tasks. In industrial automation applications, many objects have only a few stable positions. For sequences of images of dynamics scenes, the 2-d projection of the objects does not change significantly from one view to the next.

The outlines of the object often provide sufficient information for recognition. With occlusions and missing pieces, we can expect only a portion of the outline to match in two views of an object. In this paper we will concentrate on using closed outlines of objects for matching, but the main ideas should apply to matching with long curves which correspond to only a portion of an object.

A variety of contexts have been used to study the problems of matching closed boundaries. There is a series of papers [1,2,3] which report on a different techniques for recognition and motion analysis. Chow and Agarwal [1] used contour matching for studies of

simulated closed patterns using polygonal figures. McKee and Agarwal [2] looked at matching with only partial views of objects. They developed a measure to define how well two curves match. Martin and Agarwal [3] used curved boundaries in their studies of dynamic scenes. Davis [4] and Davis and Henderson [5] explored the use of relaxation matching for shape analysis. The first paper recognized islands based on their outlines, and the second combined relaxation and syntactic methods for recognition. Bhanu [6] looked at contour matching of closed contours using a relaxation technique on the piecewise linear representation of the boundary curve. Ayache [7] has given a method for accurately locating an object in a scene where there is substantial occlusion (or additional metal on the object - metal castings) using matches for key segments to force the locations of other segments. Line segment matching, without considering closed contours, has been studied by Clark et al. [8] for aerial views and Medioni [9] for stereo pairs. Here we have ignored the use of global boundary descriptions, since they tend to not work well with occlusions and missing parts.

The matching method we present here is an attempt to be more general in terms of the type of potential tasks and computationally simpler than previous methods. The previous work has shown that corresponding segments in two views can be computed reliably and used for recognition or object matching. Also, it has shown that some correct corresponding segments can be located by simple, rotation invariant, features of the segments, but these features give many extra matches. The most important consideration for matching closed contours is that the order of segments in the two views must be the same (or in strictly reverse order if mirror images are allowed). This last property was important in [2] and [3] and was also used in [4], [5], [6], and [7].

2. ALGORITHM DESCRIPTION

Rotation invariant features of line segments include the line segment length and the angle between consecutive segments. Using only these features, a given line segment in one view can readily match many segments in another view. But a consecutive sequence of border segments in one view should have few matches with consecutive or monotonically increasing sequences in the other view. Rather than comparing sequences of boundary segments, we will compute the potential matches and look for sequences.

¹This research was supported by the Defense Advanced Research Projects Agency and monitored by the Air Force Wright Aeronautical Laboratories under Contract No. F33615 82 K-1786, DARPA Order No. 3119

2.1. Boundary Descriptions

The images are of a variety of small tools on a light table for a good contrast, but the clear handles of some of the tools do not always have a sufficient contrast between the objects and the background. The objects are extracted using a simple threshold to separate them from the bright background. The boundary of the region is transformed into a sequence of straight line segments by a procedure originally developed for sequences of edge points [10]. Figure 1 shows the segment representation for two views of the nine objects. The accuracy of the line segment representation is controlled by a parameter which gives the allowable deviation of the individual points from the straight line segments. Three representations are generally computed corresponding to a deviation of 1.2 (1 has an anomalous behavior), 2 and 4. Each boundary is represented as an ordered sequence of these line segments with length and orientation for each. Figure 2 shows the two lower resolution segment representations for the image in Fig 1a. Segments are numbered in order around the boundary, thus consecutive segment numbers correspond to adjacent segments in the boundary. These segment sequences are called families. Each family corresponds to one region boundary with separate families for the interior boundaries of holes. Several families are included in the description for an image, with each family matched separately.

2.2. Initial Matching Sequences

The matching procedure considers two families at a time, one from each of two images. Each segment in the first family is compared with each in the second family to determine if they can possibly match. If they do match, then the orientation difference is stored in an array (indexed by segment numbers), called a disparity array. Possible matches are determined by comparing the segment lengths and by comparing the angles between the current segments and their respective successors. Both of these tests use a threshold chosen by the user. The length threshold is a multiplicative factor greater than 1, thus we can use the test:

$$l(1)/t < L(2) < L(1)*t$$

where $l(X)$ is the length of segment X and t is the threshold. At this point the length restriction is severe around 1.3. The angle difference threshold depends on the resolution of the line segment representation, ranging from 90° for the lowest resolution version to 45° for the highest.

Each segment will have many possible matches using these two criteria, but there should be very few cases where several consecutive segments in one image match consecutive segments in the other all with similar orientation differences. Figure 3 is an example of the data in the disparity array, values are not given, just the locations with matches. Thus, we need to find long consecutive sequences of matching segments. As the first step, we find two matches, n with m and $n+1$ with $m+1$. These two matches are used as the starting point for a simple search to find a long sequence of matches where gaps in either sequence are allowed.

2.3. Initial Matching Sequence

From the pair of initial matches we search for the next (or previous by searching backwards) matching segments using the disparity array. We look at the diagonally adjacent segments next (increment both by one) then the points off the diagonal. In the following:

X
.	Y	3	8	5	.
.	2	1	6	3	.
.	7	5	4	1	.
.	4	2	0	9	.

X is the first point located and Y the second. The search starts at 1 then continues at 2, 3, etc., after 9 we continue at 0, 1, This search continues until another matching pair is located which has an orientation difference close to that of the first pair. The threshold for close is the same as was used as a limit on the difference between the angles of segments and their successors. Gaps between the last match and the new match are filled in as matches when the new match occurs along the diagonal, i.e. both sequences skipped the same number of segments.

All possible sequences are located in the two families. If the longest sequence has enough points (5 for cases where good matches are expected, 3 as an extreme where nothing is known) then this set of matching segment pairs is used to determine the approximate transform to align the two families. If no long sequence is found, then there is no match for these two families. The transformation is one which will perfectly align a pair of matching segments. This pair is the one near the median, orientation difference (computed using the length of the segments as weights), which is longest and where both segments of the pair are close to the same length. That is, starting at the median look for the longest segment where the ratio between the segment lengths (short/long) is greater than 0.8. The best two transformations are used for the transformation refinement (along with the best transformation from the second longest matching sequence - if it is close (0.7) to the length of the longest one). This transformation only applies for mapping between the given pair of families, there will be many such transformations in the final complete match.

2.4. Transformation Refinement

Two (or possibly three) transformations for a pair of families have been generated which must be compared to select the best. With a known transformation, we use different constraints for computing matching segments. A possible match is indicated if the segments overlap in position, or nearly overlap, and the orientations are similar (90° for short segments and 20° for long ones) - after the transformation has been applied to the segment in the first image. Figure 4 shows the matches for the transformation generated by the longest and second longest sequence from the data in Figure 3.

Using each of the possible transformations, we compute the set of initial matching segments and find the longest sequence of matching segments by the same procedures

as above. A disparity value (Euclidian distance) is stored in the disparity array and is used in the search for long sequences. Since a transformation is applied to segments in the first view, the disparities should all be near zero, but the matching ideas here are more general and can be used in a stereo problem where there is no orientation transformation and similar disparities are used to separate various possibilities. The search for long sequences allows wrap around matches - if one sequence hits the end of the sequence it can start over at the beginning while the second only increments by one.

2.5 Hierarchical Matching

Multiple resolution segment representations help improve speed and accuracy. The time for matching of two families depends on the number of segments in the two families, but the alignment is best when the segments very closely follow the contours of the object. We apply the two step matching procedure to the lowest resolution representation and obtain a set of matches for many of the families. At the next higher resolution we use the known transformation as the starting point and apply the transformation refinement operation twice. (The second step primarily finds which segments match with the updated transformation rather than a more accurate transform and is primarily for display purposes.) Families which had no match at the lower resolution are processed the same as at the lowest resolution - find initial matches using the length and angle with the successor, then refine the match using position and orientation.

2.6 Matching Summary

In summary, the matching procedure can be described as two passes of two processing steps applied to each pair of families

- Pass 1, step 1 Compute likely corresponding segments by comparing all segments with all others. Use segment length and the angle between a segment and its successor to determine the match.
- Pass 1, step 2 Locate sequence of corresponding segments where the segment number increases monotonically in each image. Use these sequences to determine a good transformation to map one set into the other.
- Pass 2, step 1 Using the transformation compute a new set of likely matching segments using segment position and orientation.
- Pass 2, step 2 Locate sequences of monotonically increasing segments and determine a new transformation.

For multiple resolution data, Pass 2 is repeated as Pass 3 and 4, to determine corresponding segments at the higher resolution and yet a better transformation.

3. RESULTS

This matching program is intended to be somewhat general, it answers the two questions. Can these two sets

of segments be forced to match? What transformation will align the view in the first image with the second? Because we wish the program to work even with occlusions, the program will indicate a match when presented two partially similar objects. If the task is recognition then an evaluation of the match quality is necessary to determine which identification is best. In this paper, the results are for a basic matching task, not specifically recognition, but we do evaluate the matches and eliminate those which are much worse, based on number of matches, total disparity after transformation, total orientation differences, and total successor angle difference, than others for the same family.

The input images are of a set of tools (two pairs of pliers, two small screwdrivers, one longer one with a similar handle, one large screwdriver and one short, fat one). A mechanical pencil and a fountain pen were also included. These last two had fewer segments in the representation and, in some cases appeared as mirror images and thus did not match as reliably. Two views of all nine objects, with no occlusions, were taken, plus two more views of a subset of the objects and six other views with a variety of occlusions. The exact segment to segment match is not important since some segments only partially match, therefore, we will present the results as outlines taken from the first images transformed to line up with the objects in the second image.

Figure 1 shows the outline of the two images with all objects and no occlusions. These two images are matched with all the others (including with each other) in our experiments. Even though the images were digitized on a light table to obtain near perfect outlines, in some cases the clear handles of the screwdrivers cause problems. Figures 5-10 show some of the results - selected to show successes and problems.

3.1. Evaluation

The program locates most of the correct matches and many of the extra matches are with very similar objects. The differences between the two small screwdrivers are very minor and the handle of the long bladed screwdriver is almost the same as the two small ones, so these three often match all three possibilities (see Figs 5 and 6). In Fig. 5, there are three good matches for each of the two small screwdrivers and two for each of the larger ones. These are valid since the handles are very similar. Both of the pliers in one image match with both in the other since their handles are very similar. Many of the "incorrect" matches can be eliminated by choosing only the best match, but this also means some correct matches are missed. When two similar objects occlude each other the match for both may be with the same sequence. In Fig. 7 both pliers match with one sequence because this was the best match at the lowest resolution. The same is true of the group of three screwdrivers where all matches are with only one of the objects. Figures 8 and 9 show that, in some cases, the overlap of similar objects (the pliers) does not hurt the match. Round objects can cause difficulties (even when there are small well defined "ears") since many different rotations will give equally good matches. We show no examples here, but we encountered this problem on an earlier similar data set and mention it

as a known problem.

This matching procedure is reasonably efficient, with the total time depending on a number of factors - primarily the number of segments in the representation. For example, the matching for image 1 (Fig. 1a) with image 2 (Fig. 1b) (see Fig. 5 for the results) takes about 2 minutes 8 seconds. This includes matching at three resolutions for 9 objects in each image. Approximately 75% of the time is in computing the likely matches and 20% in searching for sequences of matches or computing the transformation. The times for the higher resolutions are not significantly greater than the lowest resolution because the matching is restricted to refining the existing matches, not searching for new ones except for the few unmatched objects. The lowest resolution match uses 116 and 119 segments from the first and second view, respectively, and compares 9 families in one view with all 9 in the second (i.e. test the match for 81 possible combinations). This requires a total of 41 seconds for all 81 comparisons. The implementation is on a PDP-10 with no special effort for low level efficiency.

4. CONCLUSIONS

We have proposed a simple relatively efficient matching procedure for comparing contours in scenes containing occlusions and multiple objects, which requires no iterative updating (relaxation). This procedure uses the order of segments around a boundary as the most important criterion for determining whether individual segments match. There are still some open problems, which are also problems for any other existing system. These include how to evaluate several different matches for use in a recognition system with a variety of similar objects. Another problem is the uniform use of holes which give multiple segment sequences for one region, and efficient handling of almost circular objects.

REFERENCES

- [1] W.K. Chow and J.K. Aggarwal, "Computer Analysis of Planar Curvilinear Moving Images," *IEEE Trans. Computer* Vol. 26, Feb. 1977, pp. 179-185.
- [2] J.W. McKee and J.K. Aggarwal, "Computer Recognition of Partial Views of Curved Objects," *IEEE Trans. Computers* Vol. 26, No. 8, Aug. 1977, pp. 790-800.
- [3] W.N. Martin and J.K. Aggarwal, "Computer Analysis of Dynamic Scenes Containing Curvilinear Figures," *Pattern Recognition*, Vol. 11, 1979, pp. 169-178.
- [4] L.S. Davis, "Shape Matching Using Relaxation Techniques," *IEEE Trans. PAMI*, Vol. 1, No. 1, Jan. 1979, pp. 60-72.
- [5] L.S. Davis and T.C. Henderson, "Hierarchical Constraint Processes for Shape Analysis," *IEEE Trans. PAMI*, Vol. 3, No. 3, May, 1983, pp. 265-.
- [6] B. Bhanu and O. Faugeras, "Shape Matching of Two-Dimensional Objects", *IEEE Trans. PAMI*, Vol. 6, No. 2, March, 1984, pp. 137-155.
- [7] N. J. Ayache, "A Model Based Vision System to Identify and Locate Partially Visible Industrial Parts," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Arlington, VA, June 1983, pp. 492-494.
- [8] C.S. Clark, D.K. Conti, W.O. Eckhardt, T.A. McCulloh, R. Nevatia, and D.Y. Tseng, "Matching of Natural Terrain Scenes," in *Proc. 5-ICPR*, Miami, FL, Dec. 1980, pp. 217-222.
- [9] G.G. Medioni and R. Nevatia, "Segment-Based Stereo Matching," *Proc. DARPA IU Workshop*, Arlington, VA, June 1983, pp. 128-136.
- [10] R. Nevatia and K.R. Babu, "Linear Feature Extraction and Description," *Comp. Graphics and Image Proc.*, Vol. 13, 1980, pp. 257-269.

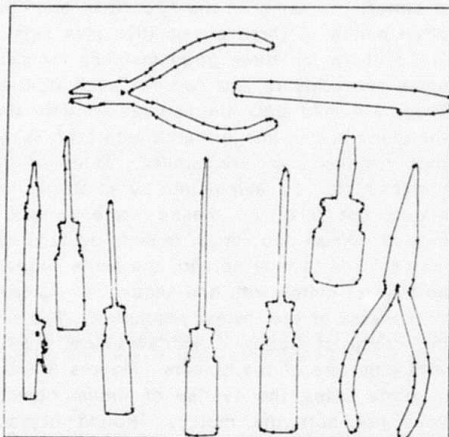


Figure 1a. Outlines of the first view of all nine objects.

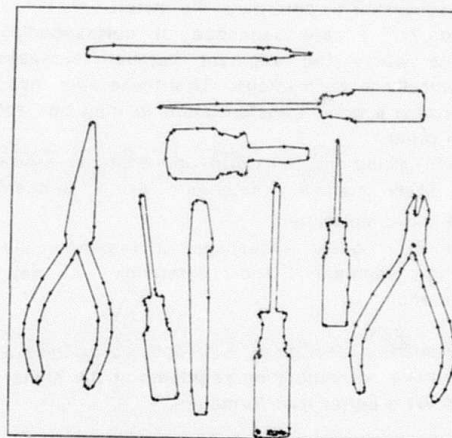


Figure 1b. Outlines of the second view of all nine objects.

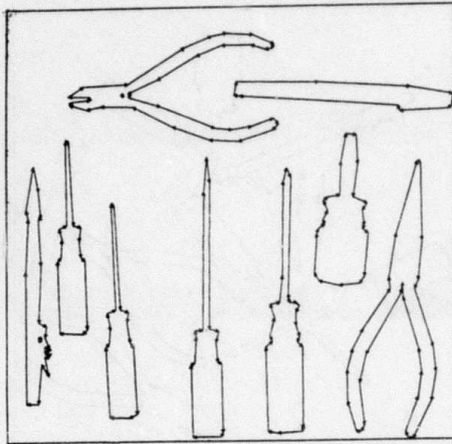


Figure 2a. Segments for a deviation threshold of 2.

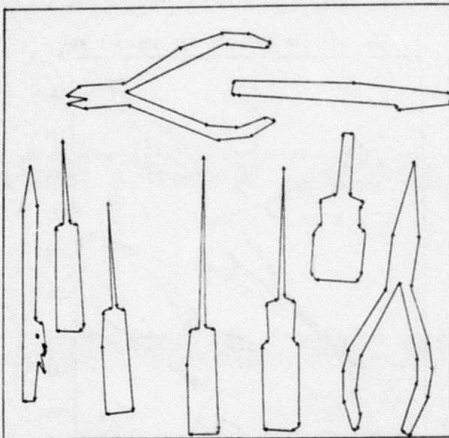


Figure 2b. Segments for a deviation threshold of 4.

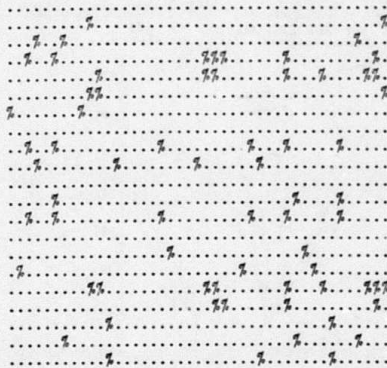


Figure 3. Matching segments for initial step. Each row corresponds to a segment in the first view, each column to a segment in the second view. The first view is for the pliers in the lower right of Fig. 1a, the second view is in Fig. 8a.

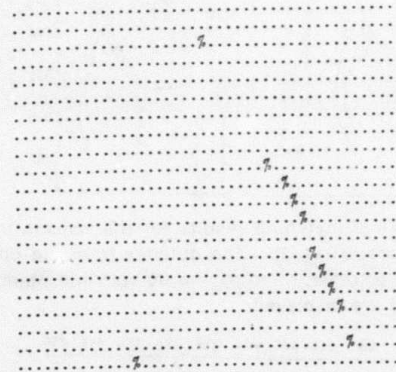


Figure 4a. The longest sequence corresponds to the incorrect pliers.

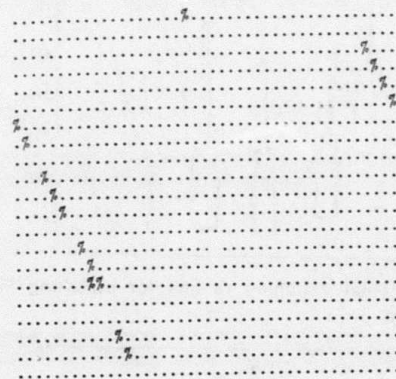


Figure 4b. The second longest sequence corresponds to the correct pliers.

Figure 4. The matching segments for the two longest sequences found in the data of Fig. 3. Transformations have been applied so that the two views are aligned.

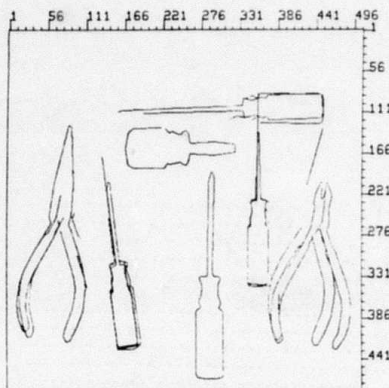


Figure 5. Final matching results for the objects in Fig. 1a with those in Fig. 1b. The outlines from the objects in Fig. 1a are transformed to line up with the objects in Fig. 1b, and the displayed.

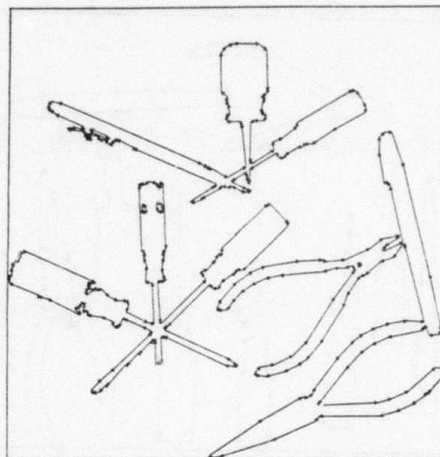


Figure 7a. Outlines of the objects in test image 9.

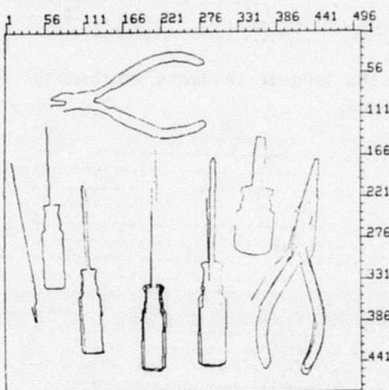


Figure 6. Final matching results for matching Fig. 1b with Fig. 1a.

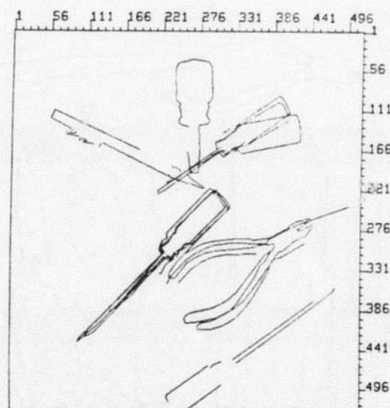


Figure 7b. Final matching results for matching Fig. 1a with Fig. 7a.

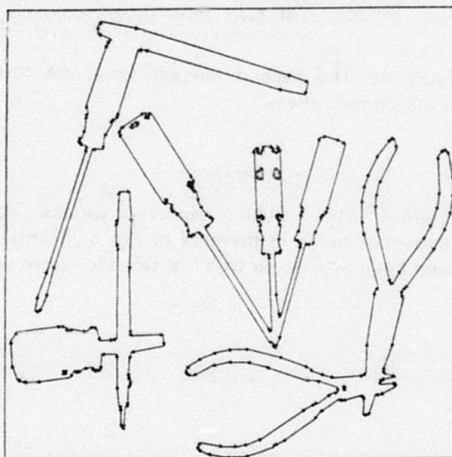


Figure 8a. Outlines of the objects in test image 6.

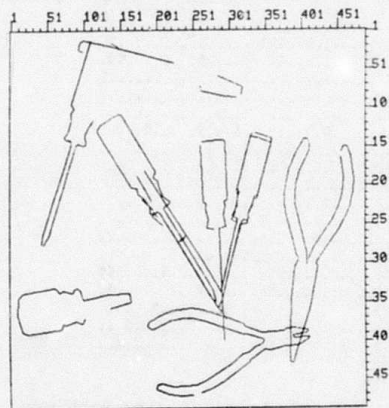


Figure 8b. Final matching results for matching Fig. 1a with Fig. 8a.

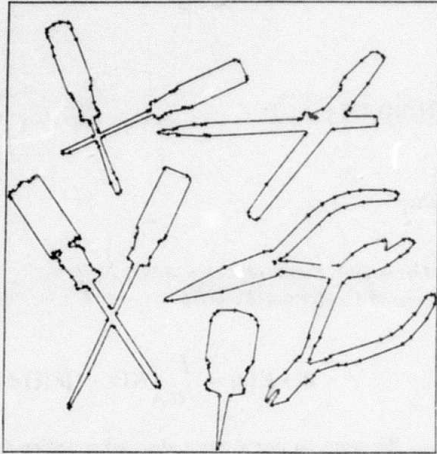


Figure 9a. Outlines of the objects in test image 8.

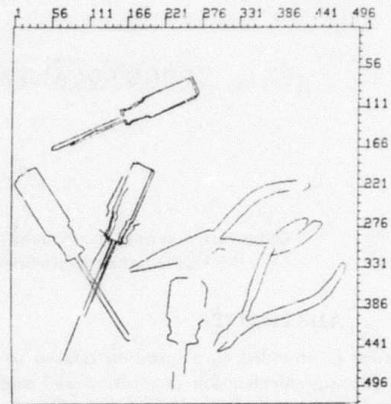


Figure 9b. Final matching results for matching Fig. 1b with Fig. 9a.

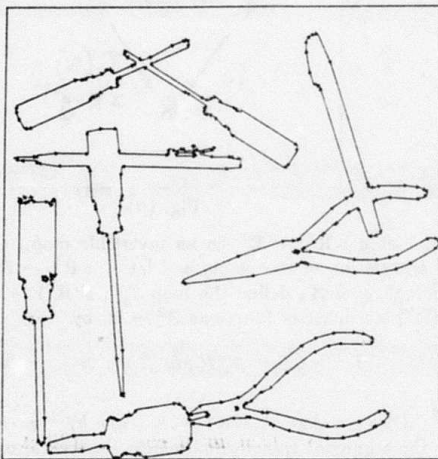


Figure 10a. Outlines of the objects in test image 7.

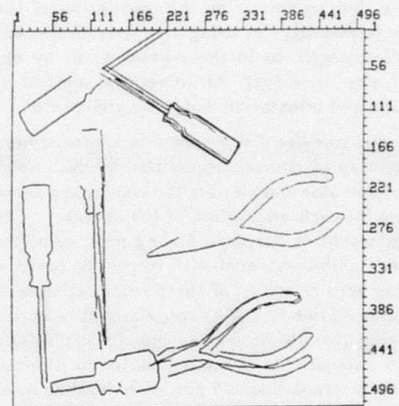


Figure 10b. Final matching results for matching Fig. 1b with Fig. 10a.

EDGE LOCALIZATION IN BOTH θ AND x

A. Peter Blicher*

Computer Science Department, Stanford University, Stanford, CA 94305
and Mathematics Department, University of California, Berkeley

ABSTRACT

A negative answer is provided to a question related to a proposal of Binford for edge localization in position and angle simultaneously by finding zero crossings of "stacked planes" of directional convolutions. Mathematical background is provided, leading to use of the inverse function theorem as the main tool of proof.

INTRODUCTION

One way to localize edges is by finding zero crossings of a convolution operator. This method yields a precise value for, say, the x -coordinate, but to determine the orientation of the edge requires further processing, e.g. using a number of oriented operators (which may disagree as to the x -position) or by observing the locus of zero crossings. An integrated method of extracting the position and orientation would be preferable.

[Binford 1981] proposes localizing edges in direction and orientation simultaneously by convolving a lateral inhibition signal with a directional operator and viewing the results as a set of "stacked planes," one for each orientation of the operator. The estimate for the edge would be based on finding maxima of the gradient of the lateral inhibition signal with respect to position and angle, by seeking zero crossings of the partial derivatives. Since all of the operations prior to finding zeroes would be implemented as convolutions, it is the zero crossings of the resulting convolutions which are sought. As ultimately stated in [Binford 1981], 2 convolutions, corresponding to 2 partial derivatives must be considered. However, it is natural to ask first whether this can be accomplished by finding the zero crossings of a single convolution. In the following, we show that this is impossible, using the inverse function theorem in what is essentially a dimensionality argument. This is why it is necessary for [Binford 1981] to require the use of 2 convolutions.

SOME MATHEMATICS OF PARAMETRIC CONVOLUTIONS

Let $F : \mathbb{R}^2 \rightarrow \mathbb{R}$ be a picture function, and $K : \mathbb{R}^2 \rightarrow \mathbb{R}$ a convolution kernel, that we also refer to as a convolution operator. The normal definition of the convolution $K * F$ is

$$K * F(x, y) = \int K(x - \xi, y - \eta) F(\xi, \eta) d\xi d\eta,$$

or, in vector notation

*This work was supported in part by ARIPA contracts MDA903-80-C-0102 and N00039-82-C-0250.

$$K * F(x) = \int_{\xi \in A} K(x - \xi) F(\xi) dA$$

We want to use a more abstract notation for this, so that we can generalize it slightly in a transparent way.

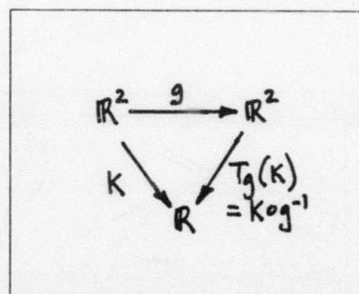


Fig. (β)

Let $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ be an invertible map, e.g. a rotation or translation of the plane, and let $K : \mathbb{R}^2 \rightarrow \mathbb{R}$. To describe "doing" g to K , define the map $T_g : \mathcal{F}(\mathbb{R}^2) \rightarrow \mathcal{F}(\mathbb{R}^2)$, where $\mathcal{F}(\mathbb{R}^2)$ is a space of functions $\mathbb{R}^2 \rightarrow \mathbb{R}$, by

$$T_g(K) = K \circ g^{-1}$$

Observe that $T_{g \circ h}(K) = K \circ (g \circ h)^{-1} = K \circ h^{-1} \circ g^{-1}$, so the argument transformations "go in reverse order" from the space transformations. Notice also the interesting fact that T_g is a linear map, even if K and g are not. Proof: $T_g(\alpha K + L) = (\alpha K + L) \circ g^{-1}$.

In particular, let G be the translation group of \mathbb{R}^2 , where $\tau_x \in G$ is defined by

$$\begin{aligned} \tau_x : \mathbb{R}^2 &\rightarrow \mathbb{R}^2 \\ p &\mapsto p + x \end{aligned}$$

$T_g(K)$ is what you get when you "move K by g ;" the right hand side of its definition shows how to calculate the new K . For a translation τ_x , $T_{\tau_x}g(K)(p) = K(p - x)$, which is often baffling for beginning students.

Define the inversion operator, ι , by

$$\begin{aligned} \iota : \mathbb{R}^2 &\rightarrow \mathbb{R}^2 \\ x &\mapsto -x \end{aligned}$$

Note that $\iota^{-1} = \iota$, and that in \mathbf{R}^2 , inversion is the same as rotation by 180° .

Using the notation for inversion and translation, the convolution formula can be rewritten

$$K * F(x) = \int K \circ \iota \circ \tau_x^{-1} \cdot F dA$$

where x is now a generic point of \mathbf{R}^2 and dA is the area measure. Note $\iota \circ \tau_x^{-1} = \tau_x \circ \iota = (\tau_x \circ \iota)^{-1}$. So, using the T notation,

$$K * F(x) = \int T_{\tau_x \circ \iota}(K) \cdot F dA$$

or, abusing the notation somewhat,

$$K * F(x) = \int T_x(K) \cdot F dA$$

We can make the notation more compact by using the L^2 inner product (\cdot, \cdot) , defined by $(g, h) = \int gh dA$:

$$K * F(x) = (T_x(K), F)$$

We can define a rotation operator, ρ , by

$$\begin{aligned} \rho_\theta : \mathbf{R}^2 &\rightarrow \mathbf{R}^2 \\ re^{i\varphi} &\mapsto re^{i(\varphi+\theta)} \end{aligned}$$

Using all the notations, we can include rotations by allowing G to be the rigid motion (Euclidean) group of \mathbf{R}^2 , and considering the functions defined by

$$\begin{aligned} S(K, F, x, \theta) &= K_\theta * F(x) = (T_{(x, \theta)}(K), F) \\ &= (T_{\tau_x \circ \iota \circ \rho_\theta}(K), F) = \int K \circ \rho_\theta^{-1} \circ \iota^{-1} \circ \tau_x^{-1} \cdot F dA \end{aligned}$$

We are interested in the function obtained from $K_\theta * F(x)$ by fixing K, F : this is a function $s : \mathbf{R}^2 \times S^1 \rightarrow \mathbf{R}$, i.e. a function of x, θ . I.e., we define s by $s(x, \theta) = S(K, F, x, \theta)$. It is the zero crossings of s which we are seeking. Let's underscore the role of the symmetry group G in the definition of s . The construction we used to define s actually defines a map $s : G \rightarrow \mathbf{R}$. In fact, for any family of K 's defined by some map $M \rightarrow \mathcal{F}(\mathbf{R}^2)$, where M is the indexing set for the family, we can define $s : M \rightarrow \mathbf{R}$. So one can easily add parameters, e.g. to allow different size operators, and this type of analysis is still applicable.

We want to show that for $s : M^3 \rightarrow \mathbf{R}$ (where M^3 is a 3-dimensional manifold) the "zero crossings" cannot be an edge locus. To do this, we will have to be more precise about what we mean by "zero crossing," and we will consider separately the cases where s is differentiable, and only continuous. The 2 cases can be analyzed independently; the continuous case subsumes the differentiable case, but since the differentiable case provides better insight, we treat it first.

To begin with, we make some observations about when s is continuous or continuously differentiable. Here is Lemma 2 of Ch. XIV §4 (p. 375) of [Lang 1969].

Lemma. Let X be a measured space with positive measure μ . Let U be an open subset of \mathbf{R}^n . Let f be a function on $X \times U$. Assume:

1) For each $y \in U$ the function $x \mapsto f(x, y)$ is in $L^1(\mu)$.

2) For each $y \in U$, each partial $D_j f(x, y)$ (taken with respect to the j -th y -variable) is in $L^1(\mu)$.

3) There exists a function $f_1 \in L^1(\mu)$ such that for all $y \in U$,

$$|D_j f(x, y)| \leq |f_1(x)|.$$

Let

$$\Phi(y) = \int_X f(x, y) d\mu(x).$$

Then $D_j \Phi$ exists and we have

$$D_j \Phi(y) = \int_X D_j f(x, y) d\mu(x).$$

The lemma permits us to conclude the following

Theorem [Lang 1969]. Let $f \in L^1$ and $\varphi \in C^r$, $r \geq 1$ with compact support. Then $f * \varphi \in C^r$ and $D^p(f * \varphi) = f * D^p \varphi$ for $p \leq r$.

Notice that this means that no matter how badly behaved f may be, $f * \varphi$ is as differentiable as φ . In particular, convolution with a C^∞ function results in a C^∞ function. In our situation, if either the picture or the convolution kernel is differentiable with respect to the parameters (we may interchange the two, allowing the symmetries to act on the picture if it suits us), then our function s , the convolution, is likewise differentiable. On the other hand it may happen that both the kernel and picture contain discontinuities, e.g. if they have steps. In that case, integration by parts yields the fact that the convolution, i.e. s , is continuous.

THE LIMITATIONS OF ZERO-CROSSINGS

Definition of zero crossings

If s is C^0 (continuous), then we will say it has a *zero crossing* at (x, y, θ) if the functions $s(\cdot, y, \theta)$ and $s(x, y, \cdot)$ have 1-dimensional zero crossings at x and θ , respectively. Colloquially, this means that the x and θ functions have zero crossings. We don't require a zero crossing in the y direction, because it may be the direction of the edge. We will say that $f : \mathbf{R} \rightarrow \mathbf{R}$ has a *1-dimensional zero crossing* at x if $f(x) = 0$, x is the only zero in some neighborhood, and f has opposite signs on opposite sides of x in such a neighborhood.

If s is C^r , $r \geq 1$, then we will say that s has a *zero crossing* at (x, y, θ) if $s(x, y, \theta) = 0$ and $D_1 s(x, y, \theta) \neq 0 \neq D_3 s(x, y, \theta)$, where D_i indicates the derivative with respect to the i -th coordinate. Thus (x, y, θ) is a regular point of s , which means that not all of its partials are 0 at that point. This implies the C^0 definition of zero crossing.

Remarks on the definition of zero crossings

The picture we are keeping in mind has the edge oriented along the y -axis. The definitions seem to single out a particular set of coordinates asymmetrically, to keep with this picture. However, the definitions really only require that the x and θ axes *not* be oriented along the edge; equivalently we could have required that *some* set of coordinate axes with these properties exist.

The condition that the 2 derivatives be nonzero when s is C^r is there for 2 reasons: First, that the definition reduce to the C^0 case, which is intuitively the meaning of "zero crossing." And secondly, to avoid degenerate cases, e.g. when the locus of zeroes is a submanifold perpendicular to the x -axis. Note that in this case, a zero can still be a regular point of s . Conversely, even if s is C^r , the C^0 condition is weaker, since e.g. it doesn't exclude tangent crossings.

Theorem. $s : \mathbb{R}^2 \times S^1 \rightarrow \mathbb{R}$ cannot have an isolated zero crossing in either of the above senses. (By *isolated* we mean there are no other zeroes in the x, θ manifold, for fixed y .) That is, edges cannot be localized simultaneously in x and θ by the zero crossings of a single (θ -parameterized) convolution operator.

Proof.

Case 1: s of class C^r , $r \geq 1$

Since (x, y, θ) is a regular point of s , the implicit function theorem applies and in some neighborhood of (x, y, θ) , $s^{-1}(0)$ is a C^r submanifold of dimension 2. The conditions on the partials guarantee that the surface is not normal to any of the x , y , or θ axes, so that for fixed y , there is a curve of (x, θ) values for which $s(x, y, \theta) = 0$, so that the zero cannot be localized in x and θ simultaneously. A more direct way to see this is to observe that what we are seeking is a function s whose zero crossings are the locus of an edge. Regarding the edge as a function $\gamma : \mathbb{R} \rightarrow \mathbb{R}^2$, it's obvious that adding orientation leads to a function $\lambda : \mathbb{R} \rightarrow \mathbb{R}^2 \times S^1$ defined by $\lambda(t) = (\gamma(t), \theta(t))$, where $\theta(t)$ is the orientation of the edge at $\gamma(t)$. Since the image of λ is 1-dimensional, we cannot hope for it to be the inverse image of a regular value of a map to the reals, since by the implicit function theorem, that must be a 2-dimensional object. But by the same token, if we have instead $s : \mathbb{R}^3 \rightarrow \mathbb{R}^2$, then one *can* try to find edges by finding $s^{-1}(0)$. QED Case 1.

Case 2: s of class C^0

We restrict attention to the function defined on the x, θ manifold, and show that every zero crossing is an accumulation point of zero crossings.

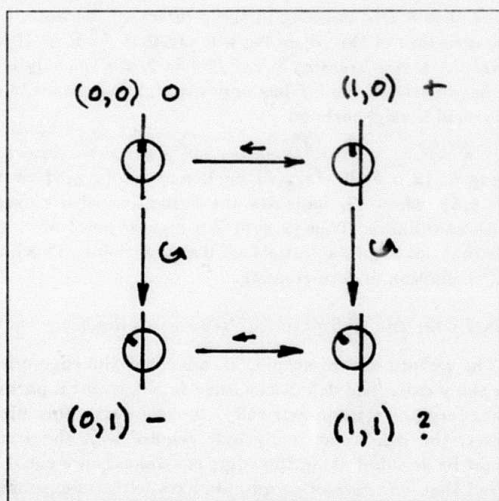


Fig. (proof1)

Look at Fig. (proof1). What it shows, schematically, is an edge operator in the vicinity of an edge, and the result of applying some motions to it. The positions are labelled on an arbitrary scale. The $(0,0)$ position is where the zero crossing is. If one assumes there are no other zeroes in some neighborhood, the indicated operations show that there are 2 ways to get to the same position of the operator with opposite signs for the result, a contradiction.

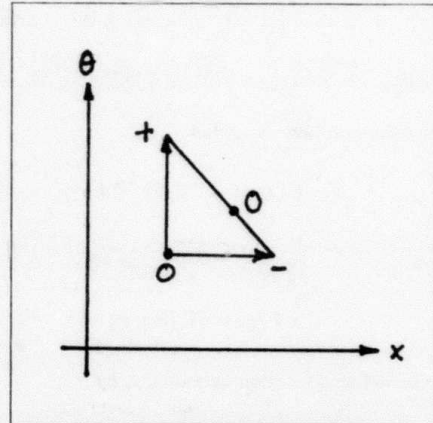


Fig. (proof2)

A more abstract picture of this is Fig. (proof2), which shows a region of the x, θ manifold near the zero crossing. In particular, we can assume without loss of generality that moving up (i.e. rotating) causes s to become $+$ (else flip the picture top for bottom), while moving right (translating) causes s to become $-$ (else flip right for left). The restriction of s to the line joining the 2 end points of these motions must have a zero, by the intermediate value theorem (see, e.g. [Rudin 1964]). QED

Acknowledgement:

This work was supported by the Defense Advanced Research Projects Agency under contract N00039-84-C-0211, Image Understanding project. I would like to thank Tom Binford for suggesting the problem this paper is concerned with, as well as for his generous help and encouragement.

REFERENCES

- [Binford 1981]
Binford, T.O., "Inferring Surfaces from Images," *Artificial Intelligence*, 17, 1981, 205-244.
- [Lang 1969]
Lang, S., *Real Analysis*, Addison-Wesley, Reading, Massachusetts, 1969.
- [Rudin 1964]
Rudin, W., *Principles of Mathematical Analysis*, McGraw-Hill, New York, 1964.

SHADING INTO TEXTURE

Alex P. Pentland
Artificial Intelligence Center, SRI International
333 Ravenswood Ave., Menlo Park, California 94025

ABSTRACT

Shape-from-shading and shape-from-texture methods have the serious drawback that they are applicable only to smooth surfaces, while real surfaces are often rough and crumpled. To extend such methods to real surfaces we must have a model that also applies to rough surfaces. The fractal surface model [Pentland 83] provides a formalism that is competent to describe such natural 3-D surfaces and, in addition, is able to predict human perceptual judgments of smoothness versus roughness — thus allowing the reliable application of shape estimation techniques that assume smoothness. This model of surface shape has been used to derive a technique for 3-D shape estimation that treats shading and texture in a unified manner.

I. INTRODUCTION

The world that surrounds us, except for man-made environments, is typically formed of complex, rough, and jumbled surfaces. Current representational schemes, in contrast, employ smooth, analytical primitives — e.g., generalized cylinders or splines — to describe three-dimensional shapes. While such smooth-surfaced representations function well in man-made, carpentered environments, they break down when we attempt to describe the crenulated, crumpled surfaces typical of natural objects. This problem is most acute when we attempt to develop techniques for recovering 3-D shape, for how can we expect to extract 3-D information in a world populated by rough, crumpled surfaces when all of our models refer to smooth surfaces only? The lack of a 3-D model for such naturally occurring surfaces has generally restricted image-understanding efforts to a world populated exclusively by smooth objects, a sort of "Play-Doh" world [1] that is not much more general than the blocks world.

Standard shape-from-shading [2,3] methods, for instance, all employ the heuristic of "smoothness" to relate neighboring points on a surface. Shape-from-texture [4,5] methods make similar assumptions: their models are concerned either with markings on a smooth surface, or discard three-dimensional notions entirely and deal only with ad hoc measurements of the image. Before we can reliably employ such techniques in the natural world, we must be able to determine which surfaces are smooth and which are not — or else generalize our techniques to include the rough, crumpled surfaces typically found in nature.

To accomplish this, we must have recourse to a 3-D model competent to describe both crumpled surfaces and smooth ones. Ideally, we would like a model that captures the intuition that smooth surfaces are the limiting case of rough, textured ones, for such a model might allow us to formulate a unified framework for obtaining shape from both shading (smooth surfaces) and texture (rough surfaces, markings on smooth surfaces).

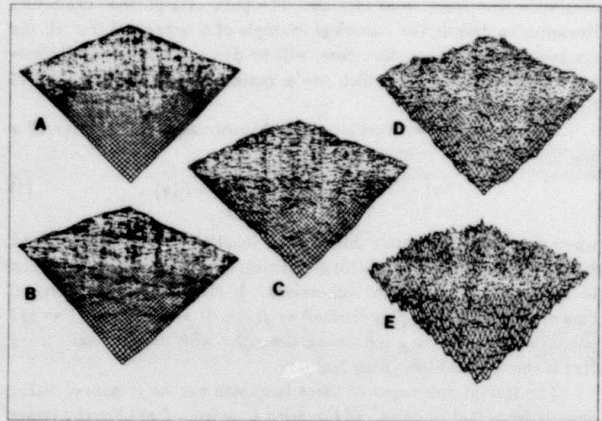


Figure 1. Surfaces of Increasing Fractal Dimension.

The fractal model of surface shape [6,7] appears to possess the required properties. Evidence for this comes from recently conducted surveys of natural imagery [6,8]. These survey found that the fractal model of imaged 3-D surfaces furnishes an accurate description of most textured and shaded image regions. Perhaps even more convincing, however, is the fact that fractals *look like natural surfaces* [9,10,11]. This is important information for workers in computer vision, because the natural appearance of fractals is strong evidence that they capture all of the perceptually relevant shape structure of natural surfaces.

II. FRACTALS AND THE FRACTAL MODEL

During the last twenty years, Benoit B. Mandelbrot has developed and popularized a relatively novel class of mathematical functions known as *fractals* [9,10]. Fractals are found extensively in nature [9,10,12]. Mandelbrot, for instance, shows that fractal surfaces are produced by many basic physical processes. The defining characteristic of a fractal is that it has a *fractional dimension*, from which we get the word "fractal." One general characterization of fractals is that they are the end result of physical processes that modify shape through local action. After innumerable repetitions, such processes will typically produce a fractal surface shape.

The fractal dimension of a surface corresponds quite closely to our intuitive notion of roughness. Thus, if we were to generate a series of scenes with the same 3-D relief but with increasing fractal dimension D , we would obtain a sequence of surfaces with linearly increasing perceptual roughness, as is shown in Figure 1: (a) shows a flat plane ($D \approx 2$), (b) rolling countryside ($D \approx 2.1$), (c) an old, worn mountain range ($D \approx 2.3$), (d) a young, rugged mountain range ($D \approx 2.5$), and, finally (e), a stalagmite-covered plane ($D \approx 2.8$).

EXPERIMENTAL NOTE: Ten naive subjects (natural-

* The research reported herein was supported by National Science Foundation Grant No. DCR-83-12766 and the Defense Advanced Research Projects Agency under Contract No. MDA 903-83-C-0027 (monitored by the U.S. Army Engineer Topographic Laboratory)

language researchers) were shown sets of fifteen 1-D curves and 2-D surfaces with varying fractal dimension but constant range (e.g., see Figure 1), and asked to estimate roughness on a scale of one (smoothest) to ten (roughest). The mean of the subject's estimates of roughness had a nearly perfect 0.98 correlation (i.e., 96% of the variance was accounted for) ($p < 0.001$) with the curve's or surface's fractal dimension. The fractal measure of perceptual roughness is therefore almost twice as accurate as any other reported to date, e.g., [13].

Fractal Brownian Functions. Virtually all fractals encountered in physical models have two additional properties: (1) each segment is statistically similar to all others; (2) they are statistically invariant over wide transformations of scale. The path of a particle exhibiting Brownian motion is the canonical example of this type of fractal; the discussion that follows, therefore, will be devoted exclusively to fractal Brownian functions, which are a mathematical generalization of Brownian motion.

A random function $I(x)$ is a fractal Brownian function if for all x and Δx

$$Pr\left(\frac{I(x + \Delta x) - I(x)}{\|\Delta x\|^H} < y\right) = F(y) \quad (1)$$

where $F(y)$ is a cumulative distribution function [7]. Note that x and $I(x)$ can be interpreted as vector quantities, thus providing an extension to two or more topological dimensions. If $I(x)$ is scalar, the fractal dimension D of the graph described by $I(x)$ is $D = 2 - H$. If $H = 1/2$ and $F(y)$ comes from a zero-mean Gaussian with unit variance, then $I(x)$ is the classical Brownian function.

The fractal dimension of these functions can be measured either directly from $I(x)$ by using* of Equation 1, or from $I(x)$'s Fourier power spectrum** $P(f)$, as the spectral density of a fractal Brownian function is proportional† to f^{-2H-1} .

Properties of Fractal Brownian Functions. Fractal functions must be stable over common transformations if they are to be useful as a descriptive tool. Previous reports [6,7] have shown that the fractal dimension of a surface is invariant with respect to linear transformations of the data and to transformations of scale. Estimates of fractal dimension, therefore, may be expected to remain stable over smooth, monotonic transformations of the image data and over changes of scale.

A. The Fractal Surface Model And The Imaging Process

Before we can use a fractal model of natural surfaces to help us understand images, we must determine how the imaging process maps a fractal surface shape into an image intensity surface. The first step is to define our terms carefully.

DEFINITION: A **fractal Brownian surface** is a continuous function that obeys the statistical description given by Equation (1), with z as

*We rewrite Equation (1) to obtain the following description of the manner in which the second-order statistics of the image change with scale: $E(\|\Delta I_{\Delta x}\| \|\Delta x\|^{-H}) = E(\|\Delta I_{\Delta x=1}\|)$ where $E(\|\Delta I_{\Delta x}\|)$ is the expected value of the change in intensity over distance Δx . To estimate H , and thus D , we calculate the quantities $E(\|\Delta I_{\Delta x}\|)$ for various Δx , and use a least-squares regression on the log of our rewritten Equation (1).

**That is, since the power spectrum $P(f)$ is proportional to f^{-2H-1} , we may use a linear regression on the log of the observed power spectrum as a function of f (e.g., a regression using $\log(P(f)) = -(2H+1)\log(f) + k$ for various values of f) to determine the power H and thus the fractal dimension.

†Discussion of the rather technical proof of this proportionality may be found in Mandelbrot [10].

a two-dimensional vector at all scales (i.e., values of Δx) between some smallest (Δx_{min}) and largest (Δx_{max}) scales.

DEFINITION: A **spatially isotropic fractal Brownian surface** is a surface in which the components of the surface normal $N = (N_x, N_y, N_z)$ are themselves fractal Brownian surfaces of identical fractal dimension.

Our previous papers [6,7] have presented evidence showing that most natural surfaces are spatially isotropic fractals, with Δx_{min} and Δx_{max} being the size of the projected pixel and the size of the examined surface patch, respectively. This finding has since been confirmed by others [8]. Furthermore, it is interesting to note that practical fractal generation techniques, such as those used in computer graphics, have had to constrain the fractal-generating function to produce spatially isotropic fractal Brownian surfaces in order to obtain realistic imagery [11]. Thus, it appears that many real 3-D surfaces are spatially isotropic fractals, at least over a wide range of scales*.

With these definitions in hand, we can now address the problem of how 3-D fractal surfaces appear in the 2-D image.

Proposition 1. A 3-D surface with a spatially isotropic fractal Brownian shape produces an image whose intensity surface is fractal Brownian and whose fractal dimension is identical to that of the components of the surface normal, given a Lambertian surface reflectance function and constant illumination and albedo.

This proposition (proved in [7]) demonstrates that the fractal dimension of the surface normal dictates the fractal dimension of the image intensity surface and, of course, the dimension of the physical surface. Simulation of the imaging process with a variety of imaging geometries and reflectance functions indicates that this proposition will hold quite generally: the "roughness" of the surface seems to dictate the "roughness" of the image. If we know that the surface is homogeneous,** we can estimate the fractal dimension of the surface by measuring the fractal dimension of the image data. What we have developed, then, is a method for inferring a basic property of the 3-D surface — i.e., its fractal dimension — from the image data. The fact that fractal dimension has also been shown to correspond closely to our intuitive notion of roughness confirms the fundamental importance of the measurement.

EXPERIMENTAL NOTE: Fifteen naive subjects (mostly language researchers) were shown digitized images of eight natural textured surfaces drawn from Brodatz [14]. They were asked "if you were to draw your finger horizontally along the surface pictured here, how rough or smooth would the surface feel?" — i.e., they were asked to estimate the 3-D roughness/smoothness of the viewed surfaces. A scale of one (smoothest) to ten (roughest) was used to indicate 3-D roughness/smoothness. The mean of the subject's estimates of 3-D roughness had an excellent 0.91 correlation (i.e., 83% of the variance accounted was for) ($p < 0.001$) with roughnesses predicted by use of the image's 2-D fractal dimension and Proposition 1. This result supports the general validity of Proposition 1.

B. Identification of Shading Versus Texture

Fractal functions with $H \approx 0$ are planar except for random variations described by the function $F(y)$ in Equation (1). If the variance of $F(y)$ is small people judge these surfaces to be "smooth"; thus, the fractal model with small values of H is appropriate for modeling smooth, shaded regions of the image. If the surface has significant local

*This does not mean that the surfaces are completely isotropic, merely that their fractal (metric) properties are isotropic.

**Perhaps determined by the use of imaged color.

fluctuations, i.e., if $F(y)$ is large, the surface is seen as being smooth but textured, in the sense that markings or some other 2-D effect is modifying the appearance of the underlying smooth surface. In contrast, fractals with $H > 0$ are not perceived as smooth, but rather as being rough or three-dimensionally textured.

The fractal model can therefore encompass shading, 2-D texture, and 3-D texture, with shading as a limiting case in the spectrum of 3-D texture granularity. The fractal model thus allows us to make a reasonable, rigorous and perceptually plausible definition of the categories "textured" versus "shaded," "rough" versus "smooth," in terms that can be measured by using the image data.

The ability to differentiate between "smooth" and "rough" surfaces is critical to the performance of current shape-from-shading and shape-from-texture techniques. For surfaces that, from a perceptual standpoint, are smooth ($H \approx 0$) and not 2-D textured ($\text{Var}(F(y))$ small), it seems appropriate to apply shading techniques.* For surfaces that have 2-D texture it is more appropriate to apply available texture measures. Thus, use of the fractal surface model to infer qualitative 3-D shape (namely, smoothness/roughness), has the potential of significantly improving the utility of many other machine vision methods.

III. Shape Estimates From Texture And Shading

The fractal surface model allows us to do quite a bit better than simply identifying smooth versus textured surfaces and applying previously discovered techniques. Because we have a unified model of shading, 2-D texture and 3-D texture, we can derive a shape estimation procedure that treats shaded, two-dimensionally textured, and three-dimensionally textured surfaces in a single, unified manner.

A. Development of a Robust Texture Measure

Let us assume that: (1) albedo and illumination are constant in the neighborhood being examined, and (2) the surface reflects light isotropically (Lambert's law). We are then led to this simple model of image formation:

$$I = \rho\lambda(\mathbf{N} \cdot \mathbf{L}) \quad (2)$$

where ρ is surface albedo, λ is incident flux, \mathbf{N} is the [three-dimensional] unit surface normal, and \mathbf{L} is a [three-dimensional] unit vector pointing toward the illuminant. The first assumption means that the model holds only within homogeneous regions of the image, e.g., regions without self-shadowing. The second assumption is an idealization of matte, diffusely reflecting surfaces and of shiny surfaces in regions that are distant from highlights and specularities [3].

In Equation (2), image intensity is dependent upon the surface normal, as all other variables have been assumed constant. Similarly, the second derivative of image intensity is dependent upon the second derivative of the surface normal, i.e.,

$$d^2 I = \rho\lambda(d^2 \mathbf{N} \cdot \mathbf{L}) \quad (3)$$

(Notation: we will write $d^2 I$ and $d^2 \mathbf{N}$ to indicate the second derivative quantities computed along some image direction (dx, dy) — this direction to be indicated implicitly by the context.)

The fractal model taken together with previous results [15], implies that on average $d^2 \mathbf{N}$ is parallel to \mathbf{N} . Consequently, if we divide Equation (2) by Equation (3) we will on average obtain the following

*Indeed, it is only in these cases that measurement noise can be reduced (by averaging) to the levels required by shape-from-shading techniques without simultaneously destroying evidence of surface shape.

relationship:

$$E\left(\left|\frac{d^2 I}{I}\right|\right) = E\left(\left|\frac{\rho\lambda(d^2 \mathbf{N} \cdot \mathbf{L})}{\rho\lambda(\mathbf{N} \cdot \mathbf{L})}\right|\right) \approx E(|d^2 \mathbf{N}|) \quad (4)$$

where $E(x)$ denotes the expected value [mean] of x . That is, we can estimate how crumpled and textured the surface is (i.e., the average magnitude of the surface normal's second derivative) by observing $E(|d^2 I/I|)$.

Equation (4) provides us with a measure of 3-D texture that is (on average and under the above assumptions) independent of illuminant effects. This measure is affected by foreshortening, however, which acts to increase the apparent frequency of variations in the surface, e.g., the average magnitude of $d^2 \mathbf{N}$. We can, therefore, obtain an estimate of surface orientation by employing the approach adopted in other texture work [5]: if we assume that the 3-D surface texture is isotropic, the surface tilt* is simply the direction of maximum $E(|d^2 I/I|)$ and the surface slant** can be derived from the ratio between $\max_\theta E(|d^2 I/I|)$ and $\min_\theta E(|d^2 I/I|)$, where θ designates the [implicit] direction along which the texture measure is evaluated. Specifically, the surface slant is the arc cosine of z_N , the z -component of the surface normal, and for isotropic textures z_N is equal to the square root of this ratio. The square-root factor is necessitated by the use of second-derivative terms.

One of the advantages of this shape-from-texture technique is that not only can it be applied to the 2-D textures addressed by other researchers [4,5] (by simply using this texture frequency measure in place of theirs†), but it can also be applied to surfaces that are three-dimensionally textured — and in exactly the same manner. This texture measure, therefore, allows us to extend existing shape-from-texture methods beyond 2-D textures to encompass 3-D textures as well.

B. Development of a Robust Shape Estimator

These shape-from-texture techniques are critically dependent upon the assumption of isotropy: when the textures are anisotropic (stretched), the error is substantial. Estimates of the fractal dimension of the viewed surface [6,7], by virtue of their independence with respect to multiplicative transforms, offer a partial solution to this problem. Because foreshortening is a multiplicative effect, the computed fractal dimension is not affected by the orientation of the surface.†† Thus, if we measure the fractal dimension of an isotropically textured surface along the x and y directions, the measurements must be identical. If, however, we find that they are unequal, we then have *prima facie* evidence of anisotropy in the surface.

This method of identifying anisotropic textures is most effective when each point on the surface has the same direction and magnitude of anisotropy, for in these cases we can accurately discriminate changes in fractal dimension between the x and y directions. When the surface texture is variable, however, this indicator of anisotropy becomes less useful. Thus, local variation in the surface texture remains a major source of error in our estimation techniques; it is therefore important to develop a method of estimating surface orientation that is robust with respect to local variation in the surface texture.

*The image-plane component of the surface normal, i.e., the direction the surface normal would face if projected onto the image plane.

**The depth component of the surface normal.

†This measure includes edge information, i.e., the frequency of Marr-Hildreth zero-crossings as we move in a given direction appears to be proportional to $E(|d^2 I/I|)$ along that direction; consider that Marr-Hildreth zero-crossings are also zero-crossings of $d^2 I/I$.

††At least not until self-occlusion effects have become dominant in the appearance of the surface.

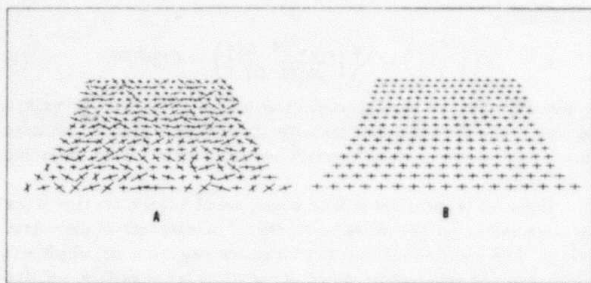


Figure 2. Variation in Local Texture (a) Compared with No Variation (b).

Such robustness can be obtained by applying regional, rather than purely local, constraints. Natural textures are often "homogeneous" over substantial regions of the image, although there may be significant local variation within the texture, because the processes that act to create a texture typically affect regions rather than points on a surface. This fact is the basis for interest in texture segmentation techniques. Current shape-from-texture techniques do not make use of the regional nature of textures, relying instead on point-by-point estimates. By capitalizing on the regional nature of textures we can derive a substantial additional constraint on our shape estimation procedure.

Let us assume that we are viewing a textured planar surface whose orientation is a 30° slant and a vertical tilt. Let us further suppose that the surface texture varies randomly from being isotropic to being anisotropic (stretched) up to an aspect ratio of 3:1, with the direction of this anisotropy also varying randomly. Such a surface, covered with small crosses, is shown in Figure 2(a); for comparison, the same surface, minus anisotropies, is shown in Figure 2(b).

If we apply standard shape estimation techniques — i.e., estimating the amount of foreshortening (and thus surface orientation) by the ratio of some texture measure along the [apparently] unforeshortened and [apparently] maximally foreshortened directions — our estimates of the foreshortening magnitude will vary widely, with a mean error of 65% and an rms error of 81%. If, however, we estimate the value α of the unforeshortened texture measure by examining the entire region, and then compare this regional estimate to the texture measure along the (apparently) maximally foreshortened direction then our mean error is reduced to 40% and the rms error to 49%.

By combining this notion of regional estimation with the texture measure developed above, i.e., $E(|d^2 I/I|)$, we can construct the following shape-from-texture algorithm that is able to deal with both smooth two-dimensionally textured surfaces and rough, three-dimensionally textured surfaces, and that is robust with respect to local variations in the surface texture.

C. A Shape Estimation Algorithm

We may construct a rather elegant and efficient shape estimation algorithm based on the notion of regional estimation and on the texture measure introduced above by employing the fact that

$$\nabla^2 I = \frac{d^2 I}{du^2} + \frac{d^2 I}{dv^2} \quad (5)$$

for any orthogonal u, v . This identity will allow us to estimate the surface slant immediately rather than having to search all orientations for the directions along which we obtain the maximum and minimum values of $E(|d^2 I/I|)$.

Let us assume that we have already determined $\alpha = \min_{\theta} E(|d^2 I/I|)$, which is the regional estimate of unforeshortened $E(|d^2 N|)$. When the estimate of α is exact, Equation (5) gives us the

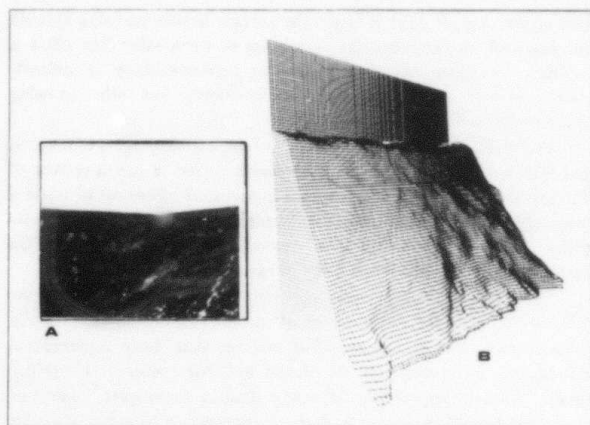


Figure 3. Tuckerman's Ravine.

result that

$$E\left(\left|\frac{\nabla^2 I}{I}\right|\right) - \alpha = \max_{\theta} E\left(\left|\frac{d^2 I}{I}\right|\right) \quad (6)$$

as the directions of maximum and minimum $E(|d^2 I/I|)$ are orthogonal.

We may therefore estimate z_N , the z component of the surface normal, by

$$z_N = \left(\frac{\beta - \alpha}{\alpha}\right)^{-1/2} \quad (7)$$

where $\beta = E(|\nabla^2 I/I|)$ and α is the regional estimate of the unforeshortened value of $E(|d^2 I/I|)$. The constant α can be estimated either by the median of the local [apparently] unforeshortened texture-measure values, or by use of the constraint that $0 \leq z_N \leq 1$ within the region. The direction of surface tilt can then be estimated by the gradient of the resulting slant field — e.g., the local gradient of the z_N values — or (as in other methods) by examining each image direction to find the one with the largest-value of the texture frequency measure. In actual practice we have found that the gradient method is more stable.

D. A Unified Treatment of Shading and Texture

The fractal surface model captures the intuitive notion that, if we examine a series of surfaces with successively less three-dimensional texture, eventually the surfaces will appear shaded rather than textured. Because the shape-from-texture technique developed here was built on the fractal model, we might expect that it too would degrade gracefully into a shape-from-shading method. This is in fact the case: this shape-from-texture technique is identical to the local shape-from-shading technique previously developed by the author [15]. That is, we have developed a shape-from-x technique that applies equally to 2-D texture, 3-D texture and shading.

As an example of the application of this shape-from-texture-and-shading technique,* Figure 3 shows (a) the digitized image of Tuckerman's ravine (a skiing region on Mt. Washington in New Hampshire), and (b) a relief map giving a side view of the estimated surface shape, obtained by integrating the slant and tilt estimates.**

*This example was originally reported in Pentland [15] as the output of a local shape-from-shading technique followed by averaging and integration. This algorithm is identical to the shape-from-texture technique described here; in fact, investigation of the shape-from-texture properties of this method was motivated by the consternation caused by this successful application of a shading technique to a textured surface.

This relief map may be compared directly with a topographic map of the area; when we compare the estimated shape with the actual shape, we find that the roll-off at the top of Figure 3(b) and the steepness of the estimated surface are correct for this surface; the slope of this area of the ravine averages 60°.

IV. Summary

Shape-from-shading and texture methods have had the serious drawback that they are applicable only to smooth surfaces, while real surfaces are often rough and crumpled. We have extended these methods to real surfaces using the fractal surface model [6,7]. The fractal model's ability to distinguish successfully between perceptually "smooth" and perceptually "rough" surfaces allows reliable application of shape estimation techniques that assume smoothness. Furthermore, we have used the fractal surface model to construct a method of estimating 3-D shape that treats shading and texture in a unified manner.

REFERENCES

- [1] H.G. Barrow and J.M. Tenenbaum, "Recovering Intrinsic Scene Characteristics From Images," in A. Hanson and E. Riseman, Eds., *Computer Vision Systems*, Academic Press, New York, New York 1978.
- [2] B. K. P. H. Horn, "Shape From Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View," A.I. Technical Report 79, Project MAC, M.I.T. (1970).
- [3] B. K. P. H. Horn and K. Ikeuchi, "Numerical Shape from Shading and Occluding Boundaries," *Artificial Intelligence*, 15, Special Issue on Computer Vision, pp. 141-184 (1981).
- [4] J. R. Kender, "Shape From Texture: An Aggregation Transform that Maps a Class of Textures Into Surface Orientation," *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo, Japan (1979).
- [5] A. P. Witkin, "Recovering Surface Shape and Orientation from Texture," *Artificial Intelligence*, 17, pp. 17-47 (1981).
- [6] A. Pentland, "Fractal-Based Description," *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)* '83, Karlsruhe, Germany, August 1983.
- [7] A. Pentland, "Fractal-Based Description Of Natural Scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, to appear September 1984.
- [8] G. Medioni and Y. Yasumoto, "A Note on using the Fractal Dimension for Segmentation," *IEEE Computer Vision Workshop*, Annapolis, MD, April 30- May 3, 1984.
- [9] B. B. Mandelbrot, "Fractals: Form, Chance and Dimension," W. H. Freeman and Co., San Francisco, California, 1977.
- [10] B. B. Mandelbrot, "The Fractal Geometry of Nature," W. H. Freeman, San Francisco, 1982.
- [11] A. Fournier, D. Fussell and L. Carpenter, "Computer Rendering of Stochastic Models," *Communications of the ACM*, vol. 25, 6, pp. 371-384, 1982.
- [12] L. F. Richardson, "The Problem of Contiguity: an Appendix of Statistics of Deadly Quarrels," *General Systems Yearbook*, vol. 6, pp. 139-187, 1961.
- [13] H. Tamura, S. Mori, and T. Yamawaki, "Textural Features Corresponding to Visual Perception," *IEEE Trans. on Sys., Man and Cyber.*, Vol. SMC-8, No. 6, pp.460-473, June 1978.
- [14] P. Brodatz, "Textures: A Photographic Album for Artists and Designers," Dover, New York, New York, 1966.
- [15] Pentland, A. P. (1984), "Local Shape Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, March 1984, pp. 170-187.

**The shape algorithm produces estimates of the surface orientation. For display purposes, these estimates were integrated to produce a relief map of the surface.

Interest Points, Disparities and Correspondence

Amit Bandyopadhyay

Computer Science Department
The University of Rochester

ABSTRACT

Matching *feature points* is a reliable way for measuring image *disparities* that arise in case of stereo or images of time varying scenes. To do this effectively we require a means of isolating the feature points. Here we describe a simple scheme to detect such interesting points in images. A correspondence algorithm then assigns matches based on local consensus. Experiments show this scheme to be robust and widely applicable to photographic images of natural scenes.

1.0 Introduction

Image *disparity* is the displacement of the image of a world point due to a shift in the camera position. Measurement of disparities used in for computing depth from stereoscopic image pairs or for analyzing motion from sequences of images of a dynamic scene. In the case of motion, given a sensor with high temporal resolution and good spatial acuity, one can compute instantaneous motion disparity using spatio temporal intensity variation and assuming smooth motion and illumination constancy. Such techniques are, however, largely untested in images of natural scenes.

Computation of disparities by matching has had some measure of success in the case of both stereo and motion. Two problems with this kind of technique are expensive computation (for correlation type of matching) and large computation time (for matching with iterative improvement or relaxation). The first problem leads to hardware complexity and the second detracts from real time applications.

The approach outlined here is an attempt to combine the benefits of both correlation type of matching and relaxation type of technique. Our algorithm does not correlate image patches, but first isolates *interest points* in the images by simple linear convolutions, which can be

implemented in parallel. Then interest points are matched to obtain disparities. The matching algorithm uses heuristics that are compatible with psychophysical observations of the visual mechanisms in man and animals (e.g. Ramachandran & Anstis 1983).

2.0 Computing Interest Points

An interest point is a point in the image (actually a small neighbourhood) that has *properties* that distinguish it from its neighbouring points. The properties in question may be simple, like gray levels, or sophisticated ones indicative of the local topography of the imaged surface. Previous approaches to finding interest points are exemplified in the work of Moravec(1977), Kitchen & Rosenfeld(1980) and Nagel(1983). The above approaches utilize operators that are nonlinear and sometimes require computation of higher order spatial derivatives of the image function. Another method is a locationwise *topological classification* of the image function. However, topological analysis is computationally expensive. Furthermore, this method has to deal with the *Hessian* of the image function at every point. Hence, it is also inherently unstable in the presence of noise. A crucial observation [Brown 1980], is that, interesting patterns in the image can be thought to have a sharply peaked *autocorrelation function*. This observation gives rise to a practical interest operator which selects image locations whose autocorrelation decays sharply with increasing eccentricity. This could be implemented with great simplicity if we could design matching templates with the above property. However no constructive algorithm exists to construct this type of template. In any case, sharply diminishing autocorrelation is a desirable property for operator templates, and it is useful to bear this fact in mind.

The location of interest points is the first step in tackling the so-called correspondence problem. Hence the operator must satisfy 3 requirements:

1. Selected points must be sparse
2. Contours must be suppressed
3. Interest points should be, stable across frames

The orthogonal decomposition technique described subsequently, is an attempt to satisfy the above requirements.

The method described here is computationally simpler than the prevalent techniques at locating interesting points in images. The *image function* is decomposed into a weighted sum of basis functions. The *central idea* being that if the basis is chosen with care, then the distribution of the respective weights indicates the nature of the image function directly. A similar approach can be seen in the literature in other image processing contexts [Frei & Chen 1977, Hueckel 1971]. A pleasing aspect of this design is that there is neuro-physiological evidence to support our approach [Hubel & Wiesel 1959]

2.1 Preliminaries

The image $I(x, y, t)$ is a three dimensional function. However, we concern ourselves with a time slice of this function at time $t = t_0$ thus obtaining a two dimensional function

$$I(x, y) = f(x, y, t_0)$$

An image vector at a location (x, y) is formed by concatenating the rows of the following 3×3 image patch

$I(x-1, y+1)$	$I(x, y+1)$	$I(x+1, y+1)$
$I(x-1, y)$	$I(x, y)$	$I(x+1, y)$
$I(x-1, y-1)$	$I(x, y-1)$	$I(x+1, y-1)$

The image vector τ belongs to a 9 dimensional *Vector Space* defined over the Real field.

$$\tau = [I_1 I_2 I_3 \dots I_9]$$

where I_1 is $I(x-1, y-1)$, I_2 is $I(x, y-1)$ and so on, alternatively

$$\tau = \sum_{k=0}^8 I_k \cdot e_k$$

where e_k is the k^{th} column of the 9×9 identity matrix.

The image vector as defined previously, is represented with respect to the basis $\{e_k\}$. The components, therefore, by themselves do not convey any information regarding the local topography of the image.

When we define the image in this manner, the operation of convolving the image with a given point spread function or correlating with a particular feature template can be expressed with respect to the vector inner product. Thus convolution becomes

$$I * g = \sum_i \sum_j \tau_i g_j$$

where g is the vector representation of the point spread function and τ is the image vector at a point.

With the above interpretation in mind we freely interchange the terms *function* and *vector* in subsequent text. More importantly, thinking of point spread functions as vectors allows us to transform the image vector into different *finite* basis space corresponding to the prototypical features that we are interested in. This transformation is wrought by a non-singular matrix T whose columns are the feature basis vectors $\{f_k\}$. Thus the image vector τ is transformed into the vector r where

$$r = T \tau = \sum_i \tau_i f_i \quad (1)$$

The purpose of this transformation is to obtain an image code whose components correspond to the degree of match between the image function and the feature functions. In general, to compute the transformed vector r from τ requires the solution of simultaneous linear equations. However, computation of the k^{th} component of r becomes simple when f_k is orthogonal to the other basis vectors in the set $\{f_i\}$. In this case, we have from equation (1)

$$r \cdot f_k = \sum_i \tau_i f_i \cdot f_k = \tau_k \|f_k\|^2$$

In particular, if the basis vectors are chosen so that they form an orthonormal set then

$$r \cdot f_k = \tau_k$$

Since the image vector is finite dimensional we can design a orthogonal basis set for the space of the image vector. In addition, this basis set is constructed in such a way that the each basis corresponds to a feature primitive. Thus decomposing the image vector in terms of the new basis would give us a new set of components (or weights) indicating the *strength* of each of the features represented by the respective basis vector.

2.2 Selection of the Feature Basis

The set of basis functions in our model is built around feature primitives like edge, maxima/minima and saddle type variation. Since the image vector is nine dimensional (i.e. the operator size is 3×3) there are nine elements in the feature basis space. There are many versions of edge masks of which the Sobel masks have been chosen. The maxima/minima feature is represented by the laplacian centre/surround operator. There are three saddle type primitives. Finally, the basis is completed by two edgeline masks and an averaging operator to capture the dc intensity. The image space is thus divided into three subspaces:

1. The Extremum subspace defined by the laplacian and saddle masks.
2. The edge subspace.
3. The average (or dc) subspace.

1	2	1
0	0	0
-1	-2	-1

1	0	-1
2	0	-2
1	0	-1

-2	1	0
1	0	-1
0	-1	2

0	-1	2
1	0	-1
-2	1	0

The Edge Basis Functions.

-1	-1	-1
-1	8	-1
-1	-1	-1

0	1	0
-1	0	-1
0	1	0

-1	0	1
0	0	0
1	0	-1

1	-1	1
-1	0	-1
1	-1	1

The Extremum Basis Functions.

1	1	1
1	1	1
1	1	1

The Averaging function.

3.0 The Matching Algorithm

The *correspondence problem* is almost universally regarded as difficult. As mentioned earlier, it arises in the measurement of image *disparities*. The problem is magnified in the case of motion since the disparity in this case is not constrained, as in the case of stereo disparity, to be parallel to a base line. The overall scheme is simple: select interest points in image frames and then decide which point from one frame matches another point from the other frame. If it is possible to obtain interest points that are sparse then correspondence is not difficult. Here sparseness means that the average disparity value is smaller than the average spatial distance between points in the same image frame. One way to obtain this, at the cost of losing accuracy, is to operate at multiple resolution. This means that the algorithm is applied to images that are bandpass filtered and sampled by different resolution grids. A better, albeit costly, way is to use correlation of image patches as a measure of the closeness of match. This again is prone to error due to noise or due to variation in the average intensity in corresponding regions in the image frames. Another source of confusion is a contour - giving rise to what Marr & Ullman call the *aperture problem* [Marr & Ullman 1981].

The outline of our algorithm is as follows :-

```
for both image frames do
  for every image location do
    begin
```

decompose the image into components with respect to normalised feature primitive basis.

decide whether to select or reject the point.

```
    end
```

*for every interest point in the first frame do
 find the point in the other frame that best matches it.*

The important parts of this algorithm are

1. The decision rule in the first loop.
2. The matching method in the second loop.

These are related since a stringent decision rule may miss many good interest points in its effort to maintain sparseness. On the other hand, if the matching algorithm is relatively sophisticated, matching labelled points, then the decision rule need not be sophisticated. Some examples of decision rules are :

- (1) Projection of the image vector onto the extremum space is greater than the projection onto the edge space. $\sum_{\text{extremum}} \tau \cdot f_k > \sum_{\text{edge}} \tau \cdot f_k$
- (2) Of all the image components in the feature space, the one corresponding to the laplacian basis is the maximum.
- (3) Either rule 1 or rule 2.
- (3) Rule 3 with the proviso that if there is a clear maximum in the projections then it is not onto one of the edge space bases.

The matching rule is then formulated according to whether the points are labelled or not. In case of unlabelled points

All neighbouring points support (vote for) a particular disparity value. Similar values support each other in a local region. Shorter length disparities are preferred. A point adopts a match for which it finds the maximum support.

The strategy is similar in spirit to the more sophisticated matchers, for instance, those using labelled points (e.g. Prager & Arbib 1983). In our case the points carry a label which is computed from the outputs of the nine basis operators. The label is a code that identifies the image point in question. Now the matcher weights the "supporting" votes according to the similarity of these codes. However, we avoid iterative refinement, which is common in similar algorithms (Barnard & Thompson 1980).

4.0 Conclusion

The experiments that we have conducted so far encourage us to pursue the orthogonal decomposition method for selection of interest points. The algorithmic design of the correspondence scheme is biologically motivated. The advantages of the method are that it is not dependent on illumination constancy, it eliminates contours and that it is implementable in parallel hardware.

References

- (1) Ballard, Dana H. & Christopher Brown. "Computer Vision". Prentice Hall, 1982.
- (2) Barnard, S.T. & Thompson, W.B. - "Disparity Analysis of Images". IEEE, PAMI - 2, 333 - 340, 1980.
- (3) Brown, C.M. - "An iterative improvement algorithm for coherent codes". Optics Comm. June, 1980.
- (4) Frei, W & C. C. Chen - "Fast boundary detection: A Generalization and a New Algorithm". IEEE Trans. Comput. 26, 988 - 998 (1977).
- (5) Horn, B.K.P. & B.G. Schunk. "Determining Optical Flow". Artificial Intelligence, 17, 185-204 (1981).
- (6) Hubel, D.H. & T.N. Wiesel. - "Receptive fields of single neurons in the cat's striate cortex". J. Physiol., vol. 148, pp 574- 591 (1959).
- (7) Hueckel M.H. - "An operator which locates edges in digital pictures" JACM, 18, 113 - 125, (1971).
- (8) Kitchen L. & A. Rosenfeld - "Gray - Level Corner Detection". TR 887. Computer Science Center, University of Maryland. (1980).
- (9) Marr, David & Shimon Ullman. - "Directional Selectivity and its Use in Early Visual Processing". Proc. Royal Soc. London, Ser. B. Vol. 211, 151-180, 1981.
- (10) Moravec, H.P. - "Towards Automatic Visual obstacle avoidance". Proc. 5th IJCAI, pp 584. (1977).
- (11) Nagel, Hans-Hellmut. - "Displacement Vectors Derived from Second order Intensity Variations in Image sequences" CVGIP, 21, 1983, 85-117.
- (12) Prager, John M. & Arbib, Michael A. - "Computing The Optic Flow: The MATCH Algorithm and Prediction" CVGIP, 21, 1983, 271-304.
- (13) Ramachandran V. S. & S. M. Anstis. - "Extrapolation of Motion Path in Human Visual Perception". Vision Research, Vol. 23, 1983, 83 - 85.
- (14) Stevens, K. A. - "Computation of Locally Parallel Structure". Biol. Cybernetics 29, 19 - 28 (1978).
- (15) Ullman S & Hildreth E. - "The Measurement of Visual Motion". In "Physical and Biological Processing of Images", Braddick O.J. & Sleight A.C. (eds.). Springer-Verlag, 1983, pp 154 - 176.



PLATE I. The interest operator applied to the photograph of a natural scene.

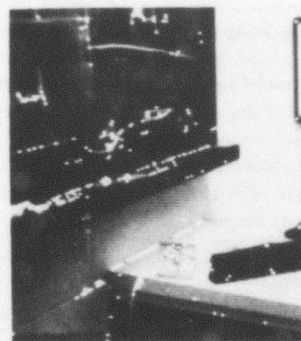


PLATE II. Interest points in an office scene.

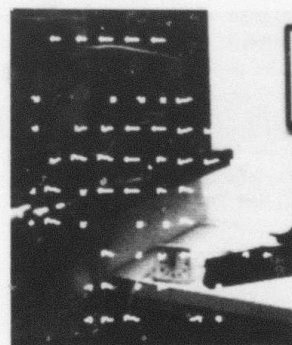


PLATE III. Correspondence established in the office scene.

A THEORY OF LINE DRAWING INTERPRETATION

Jitendra Malik and Thomas O. Binford

Computer Science Department
Stanford University, Stanford, California 94305

Abstract

This paper describes a theory for interpreting line drawings. Line Drawings are considered to be Image Structure graphs derived from the image by segmentation and aggregation operations. The goal of interpretation is to hypothesize a corresponding spatial structure. Inference Rules are derived from geometry and six additional assumptions. These rules are applicable for scenes with curved as well as planar surfaces. A control structure is sketched which works on a least commitment style of reasoning. The approach is illustrated with a worked out example.

1. Introduction.

This paper describes a theory of line drawing interpretation. Section 2 reviews past work in this and related areas and attempts to characterise the limitations of those approaches. Section 3 describes what kind of input we expect from the lower level processes. Section 4 describes the output representation. Section 5 develops our world-view—what kind of objects are permitted in the scene. Given this, we exhaustively characterise all that can possibly appear in the line drawing. Section 6 states six assumptions which seem to be the key to the process of interpretation of line drawings. Section 7 has some sample inference rules. Section 8 discusses control structure aspects. Section 9 has a solved example to illustrate the approach.

Computer implementation of this theory is under way with a program called DRISHTI.

2. Review of Past Work.

In Computer Vision, interpretation of line drawings has been taken to mean many different things.

1. Segmentation into individual bodies (Guzman [7]).
2. Identifying support edges (Falk [6], Winston [16]).
3. Line labelling using a junction catalog (Huffman [8], Clowes [1], Waltz [13]) or Gradient Space Reasoning (Mackworth [11]).
4. Finding whether the line drawing depicts an "impossible" object (Mackworth [11]).
5. Finding "optimum" three dimensional shape corresponding to a single isolated image contour (Barrow and Tennenbaum [1], Brady and Yuille [3]).
6. Inferring 3-D structure from non-accidental regularities in the image (eg skewed symmetries, parallels (Kanade [9], Binford [2], Lowe and Binford [10]).

Segmentation into individual bodies and identifying support edges is an essential component of interpretation but needs to be done in a general way as opposed to *ad hoc* heuristics which fail irretrievably in many situations.

Line labelling using a predetermined junction catalog cannot deal with scenes which violate the assumptions on which the scene is based. Also, interpretation involves more than line labelling—it includes estimates of metric information. Gradient Space, while it permits quantitative inference, suffers from another fatal flaw—as was shown quite strikingly by Draper in his thesis [5], there is a combinatorial explosion in number of 'bizarre' spatial interpretations possible for a given line drawing. By bizarre interpretations we mean physically realizable configurations which could give rise to the line drawing, but are not usually perceived by a human looking at the scene. Instead, the goal of human vision—and good machine vision—is to find one or two (in the case of certain well-known illusions like the Necker Cube) most plausible interpretations.

Detection of "impossible" objects (4) is not a primary goal of visual processing. The human system and any useful computer vision system deals with images of real objects. Line drawings can be impossible to realize as objects under "reasonable" assumptions. Failure of the interpretation process to resolve inconsistency results in the detection of impossible objects.

Present techniques used to find optimum 3-D shapes corresponding to single isolated contours ignore cues in the line drawing from other areas, interior curves, and junctions. Psychological experiments have shown the prime importance of these other cues.

The other objectives are all subsumed in our framework in a clean way as opposed to *ad hoc* techniques used in previous work.

3. Image Structure Graph

The Image Structure graph is obtained from the image by segmentation and aggregation operations. Edge finding and linking, Region growing, Corner finding, Perceptual organization are some such processes. We prefer the term *Image Structure Graph* to the term *Line drawing* as it is more suggestive and indicates a relevance to general vision. There may be multiple image structures for a portion of an image corresponding to hierarchies of description.

The nodes of the Image Structure Graph are discriminable points, curves and areas; the arcs are of two kinds—*t-arcs* (for topological relationships) and *g-arcs* (for geometric relationships). The *t-arcs* are of four kinds—*bounds* and *is-bounded-by* which have an inverse relationship, *interior* and *exterior* which have an inverse relationship. The *g-arcs* are used for representing relationships like parallelism, collinearity, symmetry etc.

These elements are found in the following way:

1. **Discriminable point.** In the image structure graph, it corresponds to either a tangent or curvature discontinuity in a piecewise smooth curve or to the intersection of two or more image curves or the termination of a curve.
2. **Curve.** A curve in the image is the output of edge finding and perceptual organisation processes.
3. **Area.** An area in the Image Structure graph is a connected subset in an image. It is defined by its boundary.

4. Spatial Interpretation Graph

The nodes of the Spatial Interpretation Graph correspond to elements of the scene in space—surfaces, curves and points. These are not the same as image structures. We use the notation $\{P_i, C_i, A_i\}$ for image points, curves and areas respectively. For scene elements we use $\{p_i, c_i, S_i\}$ for scene points, curves and surfaces. There may be scene elements, *eg* hidden surfaces, which have no corresponding elements in the Image Structure Graph.

A surface element is a surface or a surface element restricted by a boundary. A surface element is represented as a surface and a boundary or as point set operations on surface elements (union, intersection, difference). The boundary is a space curve which may be null. The surface is also typically a boundary of two volumes.

A similar representation holds for curves, surfaces, volumes, and hypervolumes. In each case, the boundary has lower dimension. For example, a curve element is a curve restricted by a boundary, i.e. a set of points, or by unions, intersections, and differences of curve elements.

In addition to arcs from a node pointing to its boundaries, there are also arcs pointing to the node/s of which it is a boundary. Also attached to each node is a set of geometrical properties like position, direction (for curves), orientation (for surface elements), curvature (for curves and surface elements). These may be specified either in an environment-

centered frame or viewer-centered frame. Heights of objects may be most conveniently expressed as above the ground plane whereas a limb boundary of a surface element is specified relative to the viewpoint (line of sight grazes the surface tangentially along the limb). These estimates may be very sketchy and can be refined by other shape-from processes.

5. Modelling the scene and the projection process

The scene consists of objects which are connected volumes bounded by smooth surfaces. A smooth surface is defined as one which has a unique tangent plane everywhere. The points on the object where there is no unique tangent plane are either isolated points, *eg* the apex of a cone, or curves—where two smooth surfaces intersect, *eg* the edges of a cube. These curves are true edges—there is a discontinuity of surface orientation across them.

We use a very simple model of illumination—diffuse lighting and nonspecular surfaces. Interpretation under more complex lighting conditions giving rise to specularities, shadows etc requires more information, *eg* brightness values, than are available in the Image Structure Graph. Incorporating these will be the subject of future work.

The various elements of the image structure graph originate in one of the following three ways:

1. **Projection of a single smooth surface.** A single smooth surface projects to an area. Parabolic lines on the surface which correspond to the transition between elliptic and hyperbolic patches, may project to significant interior curves in this area. The only other elements of the image structure graph which can result from the projection of a smooth surface correspond to the singularities of the visual mapping. These can only be folds and cusps [14]. Cusps are isolated points. Folds have traditionally been called limbs or apparent edges in vision literature.
2. **Intersection of two or more smooth surfaces.** Two intersecting surfaces give rise to a true edge. This either represents a discontinuity of surface orientation across it or a crack between two different objects. Three or more surfaces intersecting at a point in space gives rise to a junction. Edges which are space curves give rise to discriminable points corresponding to tangent or curvature discontinuities on them.
3. **Occlusion.** To catalog exhaustively all the phenomena we first observe that an *n*-dimensional manifold

can occlude only manifolds of n dimensions or less. A point can occlude another point only under a special viewpoint- occluding point, occluded point, observer in the same straight line. Similarly a curve can occlude another curve or a point only under special viewpoint. If the occluding manifold is a surface, then we have three cases. A surface occluding a point results in its absence in the image structure graph. The occlusion of a curve by a surface in the scene gives rise to (a) a T-junction and one visible segment or (b) two back to back T-junctions and two visible segments or (c) complete absence of any corresponding element in the image structure graph. The occlusion of a surface S_1 by a surface S_2 gives rise to (a) the elements associated with the occlusion of the curves of S_1 by S_2 and (b) areas corresponding to the unoccluded parts of S_1 . These are defined by the curves from (A).

Because of the finite resolution of measuring devices, solid objects may degenerate into laminae or wires in the line drawing.

6. Interpretation -Basic Approach

In order to interpret line drawings it is essential to be able to carry out simple geometric reasoning. To provide this ability a set of axioms/theorems from Euclidean Geometry and a logical inference mechanism are needed. As projection is a many-to-one mapping we need some additional assumptions to facilitate the recovery of three dimensional structure. Any of these assumptions could be violated in particular situations - they are merely very good heuristics. The fallible nature of these assumptions imposes a constraint on the logical inference mechanism - it must be nonmonotonic and allow the retraction of deductions invalid in the presence of further evidence. These assumptions fall into two classes:

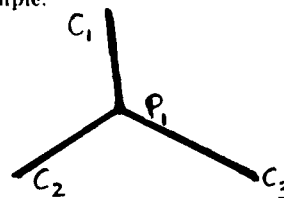
Assumptions about the Environment.

1. *Opacity.* Surfaces are opaque. This permits us to assign a unique surface to an area in the image.
2. *Solidity.* Any space curve either bounds two surfaces or is an interior curve on a surface. This encourages solid and lamina interpretations (A boundary curve of a lamina belongs to the two sides of a lamina) and discourages wires in the absence of additional evidence. $\exists S_k S_l (c_i \in S_k) \wedge (c_i \in S_l)$. If $(C_i \in A_m) \wedge (C_i \in A_n)$ S_k, S_l do not necessarily correspond to A_m, A_n .

3. *Support.* The scene consists of objects which are in physical equilibrium ie(a) the resultant of the body's weight and the reactions at the points of support are zero.(b) the net momenta about points of support are zero. From these physical conditions a set of more usable rules can be derived. It is assumed that the direction of the gravity vector is known.
4. *Frequent Structures/Relationships.* In both natural and man made scenes, certain spatial relationships like rectangularity, symmetry and verticals occur with a much higher frequency than purely by chance. The physical reasons for this are not too hard to see. This enables us to do quasi Bayesian reasoning and infer the presence of these relationships if there are supporting scenes. This is best explained with the help of an example. An orthogonal trihedral vertex (OTV) in the scene projects either to an arrow or a Y-junction which satisfies the Perkins criteria ie $\cos \theta_1 \cos \theta_2 \cos \theta_3 < 0$ where $\theta_1, \theta_2, \theta_3$ are the angles between the rays in the image. If the Perkins criteria are satisfied by the junction, then either it could be a genuine OTV or a configuration of three straight lines in space which just happens to project to an image which satisfies the criteria. The assumption of rectangularity is commonly made by humans[12]. The skewed symmetry heuristic of Kanade is another example. Proper applications of this assumption require that the a priori probability of the structure being in the scene be sufficiently high.

Assumptions about the Viewpoint.

1. *Non-accidental Viewpoint.* Interpretations which require the viewpoint to be from a set with a two-dimensional measure of zero are excluded. This assumption, originally due to Huffman, is best illustrated with an example.



Consider the Y-junction shown in the figure. The three image curves C_1, C_2, C_3 correspond to space curves c_1, c_2, c_3 which intersect in space at a vertex p_1 (inverse

of P_1). For this to be an accident the endpoints of the three curves would have to be along the same ray from the observer. This corresponds to a two-dimensional measure of zero. Alternatively we can treat this statistically. The two visible components of the difference vector are $\approx \epsilon$ each where ϵ is the size of a pixel. Given this, the maximum likelihood estimate of the third component is of the order of ϵ .

2. *Terrestrial Viewpoint.* The scene consists of objects lying on or above a ground plane being viewed by an observer above the ground plane. This assumption, obviously useful from ecological perspective for humans and other landbased animals is an important component of our perception of line drawings. It enables us to choose among Necker flip related interpretations.

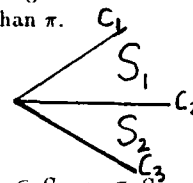
7. Some Inference Rules

We view the process of interpretation as one of constructing a *simple explanation* for the Image Structure graph. This immediately raises the questions- What is *simple*? What is an *explanation*? In our view of things an explanation is a SIG which under the projection process described in Section 5 would result in the the given ISG. The notion of simplicity is twofold (a) Occam's razor "*Entities will not be multiplied without necessity*" (b) Minimise the number of violations of the perceptual principles stated in section 6. We describe here some inference rules derived from these considerations. The Rules for interpreting the ISG can be partitioned into three groups:

1. Assigning a spatial interpretation for each node of the ISG and the t-arcs.
2. Introducing additional constraints in the SIG corresponding to the g-arcs in the ISG,
3. Making necessary geometric inferences and enforcing consistency.
1. *Instantiation Rules.* A junction, ie a discriminable point in the image corresponds to a unique point in space. A curve in the image corresponds to a unique curve in space. An area in the image corresponds to a unique surface in space. This enables us often to use the same indices for corresponding structures $P_i \rightarrow p_i$, $C_j \rightarrow c_j$, $A_k \rightarrow S_k$. This notational convention will be used in the rest of the paper, unless otherwise stated.

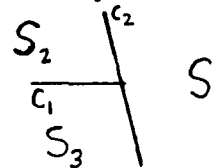
The next four rules enable us to deduce which curves belong to which surfaces.

2. *Edge Rule.* If $(C_i \in A_k) \wedge (C_i \in A_l)$, ie the image curve bounds A_k, A_l , then $(c_i \geq S_k)$ and $(c_i \geq S_l)$ ie the edge is not behind either surface. We will use \geq to denote this viewpoint-dependent relationship.
3. *Arrow Rule.* An arrow junction is defined to be one where two of the angles are less than $\pi/2$ and the third angle is greater than π .



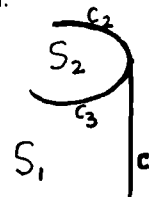
Here $c_1 \in S_1$, $c_2 \in S_1$, $c_2 \in S_2$, and $c_3 \in S_2$. This rule is interesting because to derive it, it is necessary to invoke the notion of minimum number of surfaces which can explain the junction.

4. *T-Junction.* For the T junction shown $c_2 \geq c_1$. There are two cases



1. *Occlusion T junction.* $c_2 \succ c_1 \wedge c_2 \in S_1$
2. *Object Alignment* S_1 belongs to a different object than S_2, S_3 .

5. *Limb-T Junction.*



A limb-T junction is defined by the fact that all the three curves C_1, C_2, C_3 share a common tangent at P_1 . Here it can be shown that $c_3 \in S_1$, $c_2 \in S_2$, $c_1 \in S_1$.

6. *Continuity Rules* Here we define zeroth order continuity to be coincidence in space, first order continuity to be continuity in slope and second order continuity to be continuity of curvature. Here are the rules as they apply to curves.

- 2.1 Zeroth order. Curves which intersect in the image intersect in space.
- 2.2 First order. A curve which is smooth in the image

is smooth in space.

- 2.3 Second order. Segments of the curve which have continuous curvature in the image have continuous curvature in the scene.

There are corresponding rules for surfaces. Here discontinuities can be along points or curves. An area which has no visible creases has smoothly varying curvature. This provides us with a way to estimate surfaces.

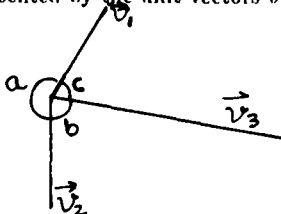
7. *Surface Estimation* A surface is estimated by first estimating relative position, orientation, and curvature at sparse locations on the surface at which curves are visible. Surfaces and curves are regarded as splines through sampled points. For example, if C_1 is straight and $C_1 \rightarrow c_1$ then assume c_1 is straight in space and that the surface c_1 lies on is singly-curved. If C_1 and C_2 are straight and intersect, then in absence of further evidence assume that the surface on which they lie is planar. This can be generalized further, as the next rule shows.
8. *Coplanarity Rules.* A set of straight lines $\{c_1, \dots, c_n\}$ is said to be coplanar iff $\exists S. \text{planar}(S) \wedge c_i \in S$. The following three rules enable us to deduce coplanarity:

$$\text{intersect}(c_1, c_2) \Rightarrow \text{coplanar}(\{c_1, c_2\})$$

$$\text{parallel}(c_1, c_2) \Rightarrow \text{coplanar}(\{c_1, c_2\})$$

$$(\text{intersect}(c_1, c_k) \vee \text{parallel}(c_1, c_k)) \wedge \text{intersect}(c_1, c_l) \\ \wedge c_k \in S \wedge c_l \in S \wedge \text{coplanar}(S) \Rightarrow \text{coplanar}(S \cup \{c_1\})$$

9. *Plane attachment* If c_j, c_k both elements of a coplanar set $\{c_1, c_2, \dots, c_n\}$ satisfy $c_j \in S, c_k \in S$ for some surface S , then c_1, c_2, \dots, c_n are all $\in S$.
10. *OTV reasoning.* Suppose that some 3-star has angles between its rays a, b and c and also that the rays are represented by the unit vectors $\vec{v}_1, \vec{v}_2, \vec{v}_3$.



Since projection is accomplished by dropping the z -component, the vectors in space must be of the form

$\vec{v}_1 + \lambda_1 \hat{z}$, $\vec{v}_2 + \lambda_2 \hat{z}$, and $\vec{v}_3 + \lambda_3 \hat{z}$ where \hat{z} is the unit vector in the z -direction.

Requiring mutual orthogonality implies that the dot products of these vectors in pairs be zero. From these conditions and some simple manipulations we can calculate the formulas for

$$\lambda_1 = \pm \sqrt{-\frac{(\cos a)(\cos c)}{(\cos b)}},$$

$$\lambda_2 = \pm \sqrt{-\frac{(\cos a)(\cos b)}{(\cos c)}},$$

$$\lambda_3 = \pm \sqrt{-\frac{(\cos c)(\cos b)}{(\cos a)}}$$

Hence solutions exist if (a) $\cos a, \cos b, \cos c$ are all non-zero and (b) either one or three of $\cos a, \cos b, \cos c$ are negative, so that the quantities under the square root sign are positive.

These results were first derived by Perkins [12]

8. Control Structure Specifications

The control problem is to be handled by a meta-level architecture. A deliberation action loop is at the heart of such an approach. There are three main reasons why we need to use meta-level rules to direct the inference process:

Reduce Search

In order to minimise search, the deductions proceed from the most reliable to the least reliable. This least commitment style of reasoning is a major characteristic of human perception as pointed out among others by Perkins[12]. We have found in our numerous hand simulations that using the following meta-heuristics, the search is negligible.

1. *Instantiate Spatial Graph from Image Graph first.*
2. *Consider neighboring vertices together and prune off inconsistencies.*
3. *Use planarity early.*
4. *Use T-junctions early.*
5. *Use geometrical rules at once if they do not require any case analysis.*
6. *Act immediately whenever an exceptional condition is flagged.*

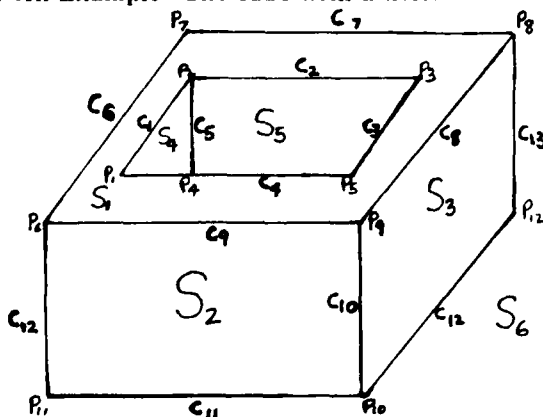
Nonmonotonic reasoning

Most of our reasoning rules have exceptions-- wires, surface marks, accidental coincidences of viewpoint etc. These are detected by either detecting contradictions or by direct inference. When contradictions are detected, a Truth Maintenance process gets into action- the offending fact/s are detected and conclusions drawn from those retracted. The Truth maintenance process can be represented by a set of meta rules which operate on the inference graph. The second way exceptional conditions are detected is by object level rules eg X-junctions denote either wires or transparency. Here again meta-level rules enable the retraction of previous conclusions.

Dealing with noisy data

The input to our program- the Image structure Graph- is obtained as the output of an edge-finding process which does not yield perfect line drawings. Lines may have missing segments, there may be more than one line corresponding to a single physical edge and so on. These get reflected in the conclusions drawn- what seems to be a surface mark may actually be a true edge with a missing segment. Again this can be handled by meta-rules like - *If you detect a surface mark, look for a continuation mark.* A model of the edge-finding process is required for this stage in order to be able to spot likely errors.

10. An Example- The cube with a hole.



Note that the labelling shown in the figure is that of the corresponding nodes in the Spatial Interpretation Graph.

1. Using the Instantiation Rule, A SIG is hypothesized with point nodes p_1, \dots, p_{12} , edge nodes c_1, \dots, c_{13} , surface nodes S_1, \dots, S_6 .
2. Using the Coplanarity rules, deduce the coplanar sets $\{c_6, c_7, c_8, c_9\}$, $\{c_9, c_{10}, c_{11}, c_{12}\}$, $\{c_8, c_{10}, c_{12}, c_{13}\}$, and $\{c_1, c_2, c_3, c_4\}$.
3. Using the arrow rule it may be deduced that
 - 3.1 $c_6 \in S_1, c_9 \in S_1, c_9 \in S_2, c_{12} \in S_2$ from P_6 .
 - 3.2 $c_{11} \in S_2, c_{10} \in S_2, c_{10} \in S_3, c_{12} \in S_3$ from P_{10} .
 - 3.3 $c_7 \in S_1, c_8 \in S_1, c_8 \in S_3, c_{13} \in S_3$ from P_8 .
 - 3.4 $c_1 \in S_4, c_5 \in S_4, c_5 \in S_5, c_2 \in S_5$ from P_2 .
4. Using the results of steps 2,3 and the Plane Attaching Inference Rule, it can be shown that

c_6, c_7, c_8, c_9 are all $\in S_1$
 $c_9, c_{10}, c_{11}, c_{12}$ are all $\in S_2$
 $c_8, c_{10}, c_{12}, c_{13}$ are all $\in S_3$
5. The T-Junction rule applied to p_4 gives either $c_4 \succ c_5 \wedge c_4 \in S_1$ or there is object alignment.
6. Consider c_3 . It is parallel to c_1 which intersects S_5 at only one point. As S_5 is planar, c_1 and c_3 are straight, it follows that $c_3 \in S_5 \Rightarrow c_3 \in S_1$ by Edge Rule.
7. As $c_1, c_3 \in S_1$ we can use the Plane Attachment rule to infer that

c_1, c_2, c_3, c_4 are all $\in S_1$
8. The Perkins criteria are satisfied at the junctions. We can apply the OTV Rule to deduce directions at the junctions and from that compute the orientations of the surfaces containing them. Support on S_6 and using the Terrestrial Viewpoint assumption, we arrive at the complete interpretation.

This example was chosen, because it was considered interesting. We have tried our approach on numerous other examples, including with curved surfaces.

Acknowledgements

This work was supported by an IBM graduate fellowship and Advanced Research Project Agency contract No N00039-84-C-0211.

References

- [1] Barrow, H.G. and J.M. Tenenbaum, "Interpreting line drawings as three-dimensional surfaces," *Artificial Intelligence*, 17 (1981), 75-116.

- [2] Binford, Thomas O., "Inferring surfaces from images," *Artificial Intelligence*, 17 (1981), 205-244.
- [3] Brady, Michael and Alan Yuille, "An Extremum Principle for Shape from Contour," *Proceedings of IJCAI-8* (Karlsruhe: August 1983), 969-972.
- [4] Clowes, M.B. "On seeing things," *Artificial Intelligence*, 2 (1971), 79-116.
- [5] Draper, Stephen W., "Reasoning about depth in line-drawing interpretation," PhD thesis, Sussex University 1980.
- [6] Falk, G., "Interpretation of imperfect line data as a three-dimensional scene," *Artificial Intelligence*, 4(2) (1972), 101-144.
- [7] Guzman, A., "Computer Recognition of three dimensional objects in a Scene," MIT Report MAC-TR-59, 1968.
- [8] Huffman, D.A., "Impossible objects as nonsense sentences," *Machine Intelligence*, 6 (1971), 295-323.
- [9] Kanade, T., "Recovery of the three-dimensional shape of an object from a single view," *Artificial Intelligence*, 17 (1981), 409-460.
- [10] Lowe, David G. and Thomas O. Binford, "The interpretation of three-dimensional structure from image curves," *Proceedings of IJCAI-7* (Vancouver: August 1981), 613-618.
- [11] Mackworth, A.K., "Interpreting pictures of polyhedral scenes," *Artificial Intelligence*, 4 (1973), 121-137.
- [12] Perkins, D.N., "Why the Human Perceiver is a bad machine," in *Human and Machine Vision*, Beck, Hope & Rosenfeld, Eds. (New York: Academic Press, 1983), 341-364.
- [13] Waltz, D., "Understanding line drawings of scenes with shadows," *The Psychology of Computer Vision*, Ed. P.H. Winston (McGraw-Hill, 1975).
- [14] Whitney, H., "Singularities of mappings of Euclidean Spaces, I: Mappings of the plane into the plane," *Ann. Math* 62 (1955) 374-410.
- [15] Witkin, Andrew P. and Jay M. Tenenbaum, "On the role of structure in vision," in *Human and Machine Vision*, Beck, Hope & Rosenfeld, Eds. (New York: Academic Press, 1983), 481-543. Abridged version appears as "What is perceptual organization for?" *IJCAI-83* (Karlsruhe, West Germany: August, 1983), 1023-1026.
- [16] Winston, P.H., "Learning Structural Descriptions from Examples," *The Psychology of Computer Vision*, Ed. P.H. Winston (McGraw-Hill, 1975).

HIERARCHICAL KNOWLEDGE-DIRECTED OBJECT EXTRACTION USING A COMBINED REGION AND LINE REPRESENTATION

by
George Reynolds
Nancy Irwin
Allen Hanson
Edward Riseman

Computer and Information Science Department
University of Massachusetts
Amherst, Massachusetts 01003

ABSTRACT

In this paper we will describe a combined region and edge representation which is used to guide the transformation of data from a low level representation to a flexible intermediate level of representation. This intermediate level of representation in turn provides information for the control of a multiresolution segmentation algorithm via multiresolution models. The system employs a goal-oriented focus of attention mechanism for directing the system to examine only areas where a coarse level segmentation yields hypotheses of object classes being sought. The experiments are carried out in the domain of digital cartography on a 4096 x 4096 pixel aerial photograph.

I. Introduction

The vast quantity of data available in the form of aerial and satellite imagery far exceeds the processing capacity of current computing environments. Efficient processing of large amounts of data requires selective focussing of expensive detailed analyses on that fraction of the image data which is likely to contain objects of interest. In many cases it might be possible to only roughly interpret or even ignore large areas of the image. Such a processing methodology implies that goal-oriented analyses, controlled by strategies which utilize both the domain knowledge as well as the goal constraints, are required. One focus of attention method involves reducing the spatial resolution of the data and using information from coarser levels to direct the processing at finer levels. We propose a hierarchical multi-resolution interpretation based on the VISIONS system architecture.

The most suitable methods for applying such selective processing to imagery of this size are the multi-resolution, or pyramid, techniques [TAN80]. Typically, from the original, large-scale, full resolution aerial image is constructed a progression of smaller and smaller images, each covering the same extent, but at successively coarser resolution. While many variations on this structure have been proposed, a pyramid is typically constructed by dividing the image up into disjoint, two-pixel by two-pixel blocks. From each block a single pixel of the image at the next coarser resolution is computed, by taking the mean or median pixel value of the block.

The underlying computational model is provided by the processing cone [RIS74, HAN78a, HAN80, GLA83], a pyramid structure implemented as a parallel array computer hierarchically organized into levels of increasing (or decreasing) spatial resolution. Level i is $2^i \times 2^i$ pixels, hence a 4096x4096 image would reside at level 12; $i=0$ corresponds to an image of one pixel and is the "top" of the cone. Processing is accomplished by executing a prototype function in parallel on windows of data at level i ; data may be processed at a given level, up the cone from finer levels to coarser levels, and projected from coarser levels to finer levels. The inter-level communication provides a mechanism by which information at coarser levels can direct more detailed processing at finer levels in the hierarchy.

This work was supported in part by the Air Force Office of Scientific Research under contract number F49620-83-C-0099 and in part by Rome Air Development Center under task number I-4-0055.

The experiments described in this paper were conducted on a 4096 x 4096 digitized aerial photograph (see Figure 1.1) with the goal of efficiently extracting instances of a variety of object classes stored in the knowledge base. This in fact represents a small image in the context of digital cartography. They were performed using a software environment which interfaces the user to virtual processing cones through LISP [KOH82, KOH83, KOH84]. This also provides a natural interface to the knowledge base and interpretation system. The preliminary results described later represent a step in the direction of developing object and scene models associated with each level in the processing cone.

II. Interpretation

The VISIONS system is an experimental testbed for investigating the construction of integrated computer vision systems for image understanding. The goal is to provide an analysis of images from segmentation through the final stages of symbolic interpretation. The output of the system is intended to be a symbolic representation of important world events depicted in the image.

The low-level segmentation system is responsible for decomposing the original image into easily manipulatable visual primitives such as regions and lines, and their attributes such as color, texture, size, shape, orientation, length, etc. While initially not dependent upon scene knowledge, the segmentation process can be made more context sensitive as the interpretation process continues via feedback from semantic processes. An interpretation is created in Short Term Memory by grouping the visual primitives in various ways and linking them to semantic labels under the constraints imposed by the knowledge base, which is referred to as Long Term Memory. This process is accomplished by applying sequences of knowledge sources which are modular process governing the transformation of data between particular levels of representation [PAR80]. One important class of knowledge sources that will be heavily used is that of a rule-oriented object hypothesis. Knowledge source application takes place under the guidance of a control strategy and extends a partially constructed interpretation of the particular image (see [HAN78c]).

II.1. Multiple Resolution Images and Models

Object models should be represented in terms of the image events (i.e., image features) which it is possible to extract from an image, as opposed to image events which it is desirable to extract from an image [MAR82]. This reasonable design observation must be adapted to the multi-resolution architectures where features of size, color/intensity, texture, shape, orientation, etc. for particular objects might only be extracted at particular levels. For example, given knowledge of the camera position in an aerial photograph, the size of a region in pixels of a given object can be predicted at the various levels of the pyramid. However, the type of averaging (or in general the type of data reduction transformation) used in constructing the higher levels of the pyramid can dramatically alter the appearance of an object and either help or hinder various algorithms for detecting these objects.

In order to reliably extract an object, the knowledge representation problem also involves specifying a description of the sequence of processes which must be applied at various levels of the pyramid. These control processes are of course not isolated since each object establishes a context which the other processes can take advantage of. The control strategies can be constructed to make use of multi-resolution object models which capture the set of visual events - i.e., different object class features - that are most appropriate at the different levels of resolution. Multi-resolution object models can be defined in terms of the line and region attributes that potentially can be extracted, the reliability and importance of these features, the processes used to hypothesize and verify objects, and the relations between objects.

II.3. Rule-Based Object Hypotheses

A simple type of knowledge source for generating hypotheses of object class labels for particular regions has been developed recently in the VISIONS environment [WEY83]. They take the form of simple rules defined in terms of ranges over a scalar feature, and complex rules defined as combinations of the output of a set of simple rules. The rules can also be viewed as sets of partially redundant features each of which defined an area of feature space which represents a "vote" for an object. The image features include color, texture, shape, size, image location, and relative location to other objects and the line features include length, orientation, contrast, width, etc.

The basic idea is to form a mapping from a measured value of the feature obtained from an image region, say f_i , into a "vote" for the object on the basis of this single feature. One approach to defining this mapping is based on the notion of prototype vectors and the distance from a given measurement to the prototype, a well-known technique which extends to N-dimensional feature space. In our case rather than using this distance to "classify", we translate it into a "vote" by defining the response of the rule as function of the distance.

In many cases, it is possible to define rules which provide evidence for and against the semantically relevant concepts representing the domain knowledge. While no single rule is totally reliable, the combined evidence from many such rules, some of which may be relatively complex and costly, should imply the correct interpretation. One function of the focussing mechanisms is to generate constraints on the application of the more costly rules.

III. Integrating Region and Line Information in an Interpretation Process

We are using a combined region and edge representation based on two low-level algorithms: the Nagin-Kohler region segmentation algorithm [NAG79, NAG82, KOH83, KOH84] and the Burns linear feature extraction algorithm [BUR84]. Both of these algorithms are briefly reviewed below. A useful characteristic of the Nagin-Kohler algorithm that will be exploited in our system is its simple localization process which can be applied selectively on various subsets of the image data. The algorithm determines feature cluster peaks in local subimages, and then forms regions based upon these histogram labels.

The Burns algorithm is a new and robust approach for extracting linear features in intensity images, including low-contrast linear features. It provides a low-level representation of intensity variations by segmenting the intensity surface into connected subsets of pixels which have similar gradient orientation; these regions act as "edge-support regions" of a linear feature, and various characteristics of the associated line or edge can be extracted from them. Thus, both regions and lines have a common pixel based representation which is a tremendous advantage for information integration.

The simultaneous use of both region and edge information permits two types of perceptual grouping processes to take place. On the one hand a region or set of regions can guide the grouping of edges, while on the other hand a line or set of lines can guide the grouping of regions. For example, the set of lines on the boundary of a region can be collected and shape descriptors obtained, or the set of short high contrast lines within the region can serve as the basis of texture measures. Similarly all the regions adjacent to a long straight line can be grouped, yielding a strategy for obtaining semantically significant collections of regions. The representation of intensity and edge information as regions facilitates the implementation of these grouping processes within the hierarchical structure of the processing cone.

III.1. Region Segmentation

The region segmentation technique employed was first developed by Nagin [NAG79] and extended by Kohler [KOH83, KOH84]. The approach is in the spirit of the Ohlander-Price algorithms [OHL78, PRI84, HAN84], but we believe it is more robust in certain situations. The algorithm involves detecting clusters in a feature histogram, associating labels with the clusters, mapping the labels onto the image pixels, and then forming regions of connected pixels with the same label. The process of global histogram labeling causes many errors to occur because global information will not accurately reflect local image events which do not involve large numbers of pixels but which nevertheless are quite clear. The Nagin algorithm overcomes this difficulty by partitioning the image into NxN subimages (usually N = 16 or 32) called sectors. The sectors are each expanded to overlap the adjacent sectors by 25%, and then the histogram segmentation algorithm is applied independently to each sector. Thus, each sector receives the full focus of the cluster detection process and many of the problems of cluster overlap and "hidden" clusters are significantly reduced. A postprocessing stage is applied to merge selected regions that have been artificially split along sector boundaries.

The partitioning of the image into sectors has an obvious weakness. If an adjacent sector has a visually distinct region which does not overlap the central sector sufficiently, it is quite possible that the cluster will be undetected in the central sector. The small region representing the intrusion into the central sector will then be lost. The Kohler algorithm improves the clustering step by adding candidate peaks from surrounding sectors to the set of peaks selected for the central sector. The augmented set of peaks forms the basis of the labeling process. The algorithm can be summarized in five steps:

- 1) Subdivide the image into sectors and select cluster labels in each expanded sector.
- 2) Analyze cluster labels from adjacent sectors for augmenting the label set.
- 3) Use the expanded set of labels to segment each sector.
- 4) Remove sector boundaries by merging similar regions.
- 5) Perform small region suppression (which often reduces the number of regions by a factor of 4).

Figure 3.1 shows the result of applying the first 3 steps above to data at level 8 in the processing cone. The level 8 data was obtained from the level 12 data by averaging over 2x2 windows from level to level. Figure 3.2 is the segmentation after elimination of sector boundaries and small region suppression.

III.2. Hierarchical Focus of Attention

The region segmentation algorithm can be applied to any subset of the set of available sectors. Starting at a coarse resolution image (level i), regions are selected which potentially correspond to a specific object modeled in the knowledge base. The sectors intersecting these regions are determined and used to select the corresponding set of sectors at the next finer level of resolution (level $i + 1$). The algorithm is applied again but only within this new subset.

Thus the processing methodology we are employing can be summarized as repeated execution of:

1. Segment the image at some level i ;
2. Select a sub-collection of the regions satisfying some object model;
3. Select those sectors which have a non-empty intersection with one of the selected regions;
4. Project those sectors to level $i + 1$ obtaining a collection of sectors at level $i + 1$;
5. Re-segment only within the sectors obtained in step 4.

For large imagery of the type considered in this paper, the computational advantage of this approach is significant. For example if we assume that even 2/3 of the possible sectors are used at each level, then only 1/5 of the image is being examined 4 levels down. In the case that only 1/4 of the sectors are selected, only 1/256th of the image is being searched 4 levels down. In addition, the knowledge base can be structured to be level dependent such that at finer levels of resolution the description of objects, and hence the inferring process, will be more complicated, while at the same time the system is looking at only a small fraction of the image. With these types of strategies the computational complexity of the process can be kept within reasonable bounds.

III.3. Finding Buildings

In the aerial image of Figure 1.1, many of the large buildings appear as small, bright regions at coarse levels of resolution. Although such a simple selection rule allows the system to hypothesize likely locations of buildings, the evidence is too unreliable to permit actual labeling of

individual buildings. The coarsest level where reliable shape features can be extracted is level 10 (1024×1024), although to exhaustively search level 10 for buildings would be very expensive. Rather, the system starts building candidate regions at a coarser level (here, level 8) and builds a mask which specifies likely areas. The mask is refined level by level until more expensive features can be extracted at level 10, but now focussing only on the likely areas.

Using the level 8 Nagin-Kohler segmentation (Figure 3.2), size and brightness features were calculated for each region. Figure 3.3 shows regions whose average brightness is greater than 160, regions whose size is less than 70 pixels, and the intersection of both sets of regions. By this criteria, any sector intersecting a bright, small region was marked. The resulting mask is shown in figure 3.4. Using the pyramid structure to project the mask, the segmentation algorithm was applied under this mask at level 9 (figure 3.5). Note that the intensity range in the rule at one level could be different than those at other levels, and that the size range has a natural scaling to the lower levels. The features of intensity and size were calculated for this new segmentation, and the upper left hand quadrant is shown in figure 3.6. A new mask is created by selecting a sector if it intersects a region satisfying a size and intensity constraint (figure 3.7). This mask is projected to level 10 and the Nagin-Kohler algorithm is applied. Again, regions are selected on the basis of size and intensity criteria and the result is displayed in figure 3.8.

III.4. Linear Feature Extraction

Intensity changes in images are due in part to reflectance, depth, orientation and illumination discontinuities and the interpretation of such changes is fundamental to computer vision. The organization of significant local intensity changes - called "edges" - into more global coherent events - called "lines" or "boundaries" - is an early but important step in the transformation of the visual signal into useful intermediate constructs for interpretation processes. Despite the vast amount of research appearing in the literature, even the extraction of linear boundaries has remained a difficult problem in many image domains. There are two goals of the approach presented here: a) the development of mechanisms for extracting linear features from complex images, including low contrast lines; and b) the construction of an intermediate representation of edge/line information which allows high-level mechanisms efficient access to relevant image events. A more detailed presentation can be found in [BUR84].

The linear feature algorithm is based on the contention that edge magnitude (local intensity change) is inherently unreliable as a local estimate of the meaningfulness of edge events. Some of the problems with using edge magnitude include those caused by:

- a) discrete sampling - sharp intensity changes involve "mixed pixels" which result in the full edge contrast being distributed across several pixels;
- b) wide gradients - total edge contrast may be distributed across many pixels so that no optimum edge operator size can be fixed a-priori (and there are no uniformly acceptable mechanisms for combining responses from different operator sizes and placements);
- c) world events - a change in background or foreground objects along an edge may cause the local magnitude estimates and total edge contrast to vary in arbitrary ways.

If the entire set of pixels associated with a particular linear feature were determined, then the decision about the line could be made globally as opposed to locally; in purely local decisions, weak or noisy edge information could confuse the process which locates the edge. While we cannot provide detailed discussions of the issues here, the use of gradient orientation as an early organizing criterion appears to be effective in overcoming these problems.

The general approach to extracting linear image events is to group the pixels into "edge support" regions of similar gradient orientation, and then to treat each region as a representation of a straight line segment. Note that every intensity variation (including very low magnitude changes) will initially be extracted as a line segment. During the interpretation of these events, adjacent low contrast support regions can be grouped into homogeneous regions and filtered so that they are not viewed as weak linear events.

There are four basic steps in extracting linear image events:

- a) Group Pixels into Edge-Support Regions. Sets of pixels are grouped based upon similarity of gradient orientation. This allows data directed organization of edge contexts without commitment to masks of a particular size.
- b) Approximate the Intensity Surface by a Weighted Planar Fit. A plane is fit to the intensity surface of the set of pixels in each support region. The fit is weighted by the gradient magnitude associated with the pixels so that the important intensity changes will dominate.

c) Extract Attributes from the Edge-Support Context. A straight line and a set of features is then extracted from the support region and the planar fit of its associated intensity surface. The attributes extracted include length, contrast, sharpness (width), location, orientation, and straightness.

d) Filter to Extract Straight Lines and Texture. Various image events, such as long lines, high contrast short lines, low contrast short lines, and lines at particular orientations, can then be extracted by filtering on the desired attributes.

An example of this algorithm applied to the level 8 image (256 x 256) with all lines of length greater than 2 pixels (and no filtering for contrast) is shown in figure 3.9.

III.5. Combining the representations

The region and edge segmentations complement each other in several ways. The region segmentation algorithm segments an image on the basis of homogeneity in intensity, color, or texture features. In this paper we are only using intensity for the region segmentation, thereby producing what will be called intensity regions. The edge algorithm defines regions, usually several pixels wide, which support edges. The most important point, however, is that each of the two segmentation processes results in a pixel-based representation. The natural duality between regions and their boundary lines can be exploited in a straight forward manner, since the edge-support region associated with a linear feature should overlap the regions on either side of the region boundary. This means that a set of edge support regions can be expected to be found superimposed on the boundary of an intensity region. Conversely, a set of adjacent intensity regions can be associated with an edge support region of a linear feature. This grouping of regions in one segmentation based on information contained in the other is performed by tracing the boundary of a selected region in the first while collecting the regions encountered in the second. In a parallel architecture such as the processing cone the set of related regions and lines can be collected in parallel.

There are several other advantages to using the combined line and region information. The lines produced by edge support regions tend to be much more accurate than the boundaries of the region segmentation due to the global nature of determining the line placement; eg. this overcomes local problems of mixed pixels. Thus the limits of regions can be determined more accurately. The overlap of short high-contrast lines with regions also provides a texture attribute for regions. Finally, a set of edges can be associated with a region and descriptors pertinent to shape can be extracted from these lines. These observations led to improved algorithms for the detection of buildings.

Let us consider one algorithm for verifying buildings based upon the line and region information. One such algorithm traces boundaries of selected intensity regions to gain from the corresponding edge segmentation the collection of edges surrounding each region. Lines are selected which lie on the boundaries of regions hypothesized to be buildings based upon size and intensity information. Results of the linear feature algorithm on a subimage for which the building hypothesis is to be verified is shown in figure 3.10.

Rectangularity is a property which can be expected of many buildings and several measures can be applied to test the set of lines selected for this and a variety of other geometric properties. For, example the orientation of the lines can be histogrammed as a weighted contribution of the length of the lines. One simple measure is the degree to which the lines group into two colinear sets at each of two orientations that are 90 degrees apart. Figure 3.11 highlights regions whose average orientations between the two clusters is within .15 radians of a right angle. The labelling of some buildings provides a context for finding others. It also reduces the search space for the processing of algorithms which do require the finest level of resolution such as measuring height to the best possible accuracy in stereo pairs and determining height

from shadow length.

III.6. Finding Runways

Runways are an example of a large, clear, linear structure whose location may be hypothesized at a very coarse resolution. The identifying feature is a set of long lines appearing close together with similar slope. The location of such a group provides the system with a starting point, but does not locate the full extents of the runway since fragmented sections of lines at the ends of the runway may fall below length thresholds. If, on the other hand, intensity regions could be identified the homogeneity of brightness would more reliably give us the full extent of the runway. The idea of the method which follows is to trace the boundaries of the selected edges to identify the pertinent intensity regions. These regions can be used to build a mask to direct further processing.

Figure 3.12 shows the image reduced to level 7 using 2x2 averaging up the cone. The line algorithm is applied and the result of filtering the edges on contrast and length are shown in figure 3.13 and 3.14.

The edge edge representation includes the position ρ and orientation θ of each line. This facilitates the grouping of lines via clusters in Hough space. The lines remaining after the filtering shown in figure 3.14, were mapped to Hough space. The largest peak in this histogram was selected and the edge support regions of that cluster were identified. These selected edge support regions, projected from level 7 to level 8, are highlighted in Figure 3.15. The adjacent regions obtained from the region segmentation applied to the intensity image are highlighted in Figure 3.16. We now apply the focus of attention mechanism to the regions, selecting the sectors are shown in figure 3.17. These sectors are projected to level 9 and the region algorithm is applied within these sectors. The result is shown in figure 3.18.

IV. Conclusions and Future Directions

We have proposed an architecture for interpreting large images based on hierarchical segmentation and interpretation processes functioning under focus of attention mechanisms. The experiments discussed in Section III demonstrate the potential effectiveness of integrating two particular complementary segmentation algorithms into a multi-resolution structure. They have shown how a range of object hypothesis rules can be used to guide and constrain the process of locating instances of the objects. The rules themselves are hierarchical and the strategies for applying them are a function of the features which can be reliably extracted at the various image resolutions.

The selection of candidate regions for examination at a higher resolution was accomplished by choosing all regions which satisfied a set of object dependent constraints on region and line attributes. In general, the results from such a simple rule will be unreliable and prone to error. Image interpretation is implicitly involved with the problems associated with combining information from multiple sources of knowledge. Any perceptual system which utilizes processed sensory data must recognize the fact that to varying degrees the information will be imperfect and prone to error. With this in mind we are developing mechanisms for evidential reasoning and inferencing under uncertainty [LOW82, WES82] in order to construct more reliable focussing sets.

Some of the limitations of inferencing using Bayesian probability models are overcome using the Dempster-Shafer formalism for evidential reasoning, in which an explicit representation of partial ignorance is provided [SHA76]. The inferencing model allows "belief" or "confidence" in a proposition to be represented as a range within the [0,1] interval. The lower and upper bounds represent support and plausibility, respectively, of a proposition, while the width of the interval can be interpreted as ignorance. Wesley [WES83] is extending this approach to the problem of distributed control of a set of knowledge sources which can be applied to examine particular concepts in long-term memory.

The object hypothesis rules described in the last section can be applied to both region and line attributes provided by our intermediate level of representation. A set of KSs associated with objects can be constructed so that when executed they will input evidence to the knowledge network as support or refutation of propositions. The inference engine will then be turned on to indirectly update belief in other propositions based upon the implications of the direct evidence.

Once the inference network is fully integrated, we expect the hierarchical segmentation and interpretation process to operate as follows. First the local histogram-based region segmentation and the linear feature extraction algorithm are applied at a coarse level of resolution. Knowledge sources in the form of object hypothesis rules are then applied to region and line attributes at that level. The output of these rules is converted to a form appropriate for input into the inference network of long-term memory. The inferencing process is then invoked and each region yields a support and plausibility (i.e., a range of belief) that it is a candidate region for one of the goal objects. The region and line segmentation algorithms are then applied at a finer level of resolution, but only on the candidate regions which have high support. At this finer level of resolution the representation of the object is of a different form and may involve more expensive object rule combinations of the region and line attributes, but applied only to a small subset of the image. The process is recursively applied to finer levels of resolution. Other ongoing research uses the inference net to focus the system within the various feature spaces that are active and that work will also be reported in the future.

References

- [BUR84] Burns, B., Hanson, A., Riseman, E., "Extracting Linear Features", to appear in *7th IJPR*, Montreal, 1984.
- [DEM67] Dempster, A., "Upper and Lower Probabilities Induced by a Multivalued Mapping," *Annals of Mathematical Statistics*, 38, pp. 325-339, 1967.
- [DEM68] Dempster, A., "A Generalization of Bayesian Inference," *Journal of the Royal Statistical Society, Series B*, 30, pp. 205-247, 1968.
- [GAR81] Garvey, T., Lowrance, J. and Fischler, M., "An Inference Technique for Integrating Knowledge From Disparate Sources" *Proc. 7th IJCAI*, pp. 319-325, 1981.
- [GLA81] Glazer, F., "Computing Optic Flow", *Proc. IJCAI-7*, August 1981, pp. 644-647.
- [GLA83] Glazer, F., Reynolds, G., Anandan, P., "Scene Matching by Hierarchical Correlation", *Proc. of the IEEE Computer Vision and Pattern Recognition Conference*, June 1983, pp. 432 - 440.
- [HAN74] Hanson, A. and Riseman, E., "Preprocessing Cones: A Computational Structure for Scene Analysis", COINS Technical Report 74C-7, University of Massachusetts, September 1974. Also in [TAN80].
- [HAN78a] Hanson, A. and Riseman, E., "Segmentation of Natural Scenes", in *Computer Vision Systems* (A. Hanson and E. Riseman, eds.), Academic Press, 1978, pp. 129-163.
- [HAN78b] Hanson, A. and Riseman, E., *Computer Vision Systems* (Eds.), Academic Press, New York, 1978.
- [HAN78c] Hanson, A. and Riseman, E., "VISIONS: A Computer System for Interpreting Scenes", in *Computer Vision Systems* (A. Hanson and E. Riseman, eds.), Academic Press, 1978, pp. 303-333.
- [HAN80] Hanson, A. and Riseman, E., "Processing Cones: A Computational Structure of Image Analysis", in *Structured Computer Vision* (S. Tanimoto, ed.), Academic Press, New York, 1980. Also COINS Technical Report 81-38, University of Massachusetts, December 1981.
- [HAN84] Hanson, A., Riseman, E., Nagin, P., "Authors Reply" to [PRI84], *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, March 1984, p. 249.
- [KOH82] Kohler, R. and Hanson, A., "The VISIONS Image Operating System", *Proc. of 6th International Conference on Pattern Recognition*, Munich, Germany, October 1982.
- [KOH83] Kohler, R., "Integrating Non-Semantic Knowledge Into Image Segmentation Processes", Ph.D. thesis, University of Massachusetts, September 1983.
- [KOH84] Kohler, R., "Integrating Non-Semantic Knowledge Into Image Segmentation Processes", COINS Technical Report 84-04, University of Massachusetts, March 1984.
- [LOW82] Lowrance, J., "Dependency-Graph Models of Evidential Support", Thesis, University of Massachusetts, Amherst, MA., 1982.
- [LOW83] Lowrance J., Garvey T., Evidential Reasoning: An Implementation for Multisensor Integration, SRI Technical Note 307, Dec. 1983.
- [MAR82] Marr, D., *Vision*, W. H. Freeman, San Francisco, 1982.
- [NAG78] Nagao, M. and Matsuyama, T., *A Structural Analysis of Complex Aerial Photographs*, Plenum Press, New York, 1980.
- [NAG79] Nagin, P., "Studies in Image Segmentation Algorithms Based on Histogram Clustering and Relaxation", COINS Technical Report 79-15, University of Massachusetts, September 1979.
- [NAG82] Nagin, P., Hanson, A. and Riseman, E., "Studies in Global and Local Histogram-Guided Relaxation Algorithms", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-4, No. 3, May 1982, pp. 263-277.
- [OHL75] Ohlander, R., "Analysis of Natural Scenes", Ph.D. Dissertation, Carnegie-Mellon University, April 1975.
- [OHL78] Ohlander, R., Price, K. and Reddy, R., "Picture Segmentation Using a Recursive Region Splitting Method", *Computer Graphics and Image Processing*, Vol. 8, pp. 313-333, 1978.
- [OHT80] Ohta, Y., "A Region-Oriented Image-Analysis System by Computer", Ph.D. Thesis, Kyoto University, Department of Information Science, Kyoto, Japan, 1980.
- [PAR80] Parma, C., Hanson, A. and Riseman, E., "Experiments in Schema-Driven Interpretation of a Natural Scene", COINS technical report number 80-10, University of Massachusetts, 1980. Also in *Nato Advanced Study Institute on Digital Image Processing* (R. Haralick and J.C. Simon, eds.), Bonas, France, 1980.
- [PRI77] Price, K. and Reddy, R., "Change Detection and Analysis in Multispectral Images", *Proc. of 5th International Joint Conference on Artificial Intelligence*, Cambridge, 1977, pp. 619-625.

- [PRI84] Price, K., "Image Segmentation: A Comment on "Studies in Global and Local Histogram-Guided Relaxation Algorithms", *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, March 1984, pp. 247-248.
- [RIS74] Riseman, E. and Hanson, A., "The Design of a Semantically Directed Vision Processor", COINS Technical Report 74C-1, University of Massachusetts, January 1974.
- [SHA76] Shafer, G., *A Mathematical Theory of Evidence*, Princeton University Press, 1976.
- [TAN80] Tanimoto, S. and Klinger, A., *Structured Computer Vision*, Academic Press, New York, 1980.
- [WES82] Wesley, L., "Control as an Inference Process: A model for Making Resource Allocation Decisions in the VISIONS System", COINS technical report in preparation, University of Mass, 1982.
- [WES82] Wesley, L., Hanson, A., "The Use of an Evidential-Based Model for Representing Knowledge and Reasoning about Images in the Visions System", in *Proc. of the Workshop on Computer Vision Ringe*, New Hampshire, Aug. 23-25, 1982, IEEE Computer Society Press.
- [WES83] Wesley, L., "Reasoning About Control: An Evidential Approach", SRI Tech. Memo, to appear.
- [WEY83] Weymouth, T., Griffith, J., Hanson, A., Riseman, E., "Rule Based Strategies for Image Interpretation", *Proc. AAAI-83*, August 1983.

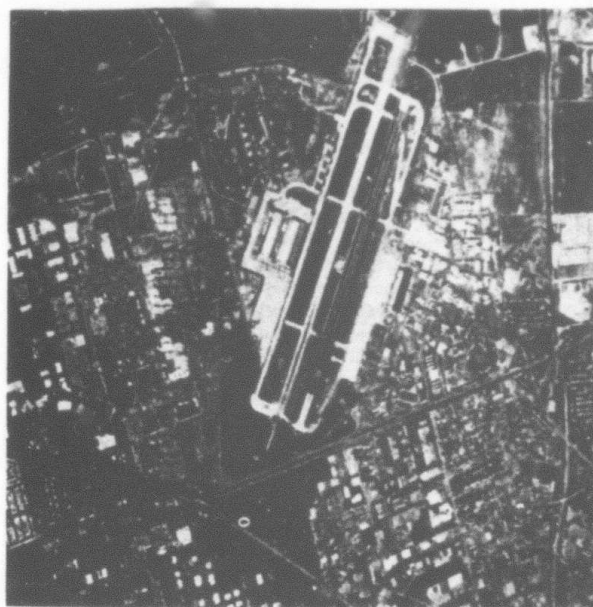


Figure 1.1.

4096 x 4096 image reduced to level 9 (512 x 512).

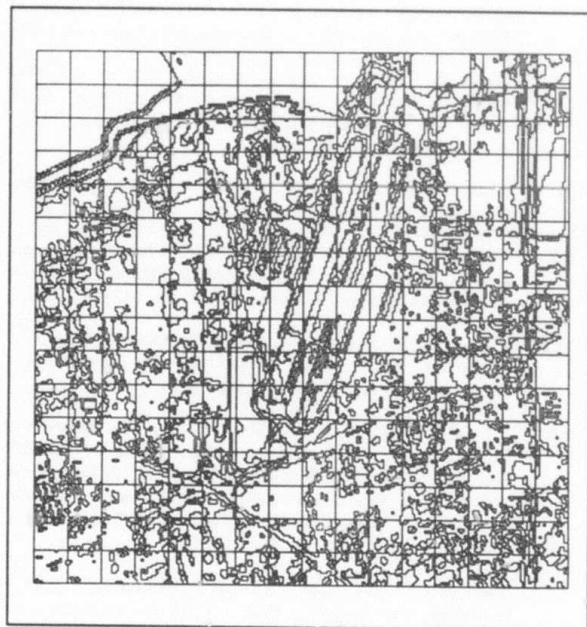


Figure 3.1.

Segmentation of each sector using the expanded set of labels.

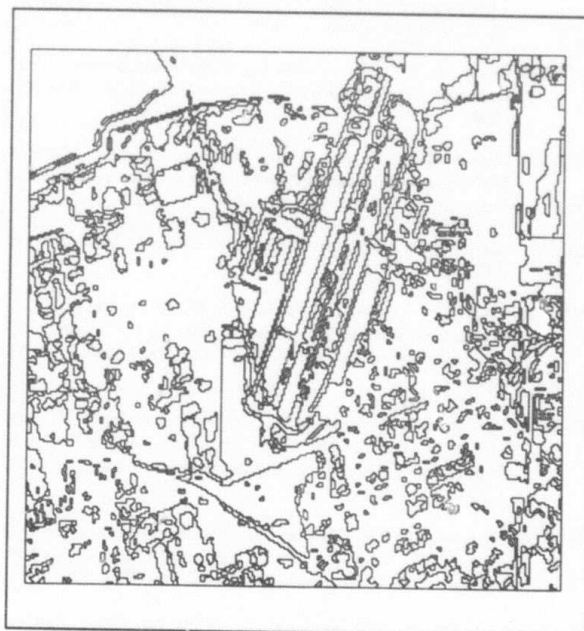


Figure 3.2.

Segmentation after removing sector boundaries and elimination of small regions.

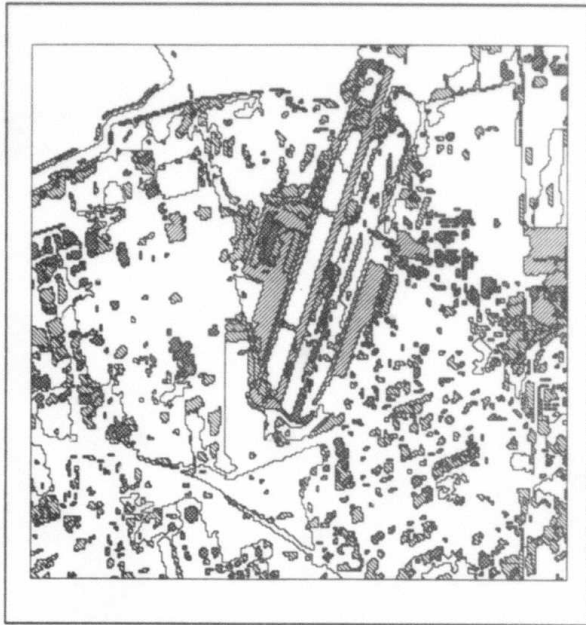


Figure 3.3.

Level 8 segmentation with small regions filled with \\//, bright regions filled with ///. Thus bright small regions are displayed as cross-hatched areas.

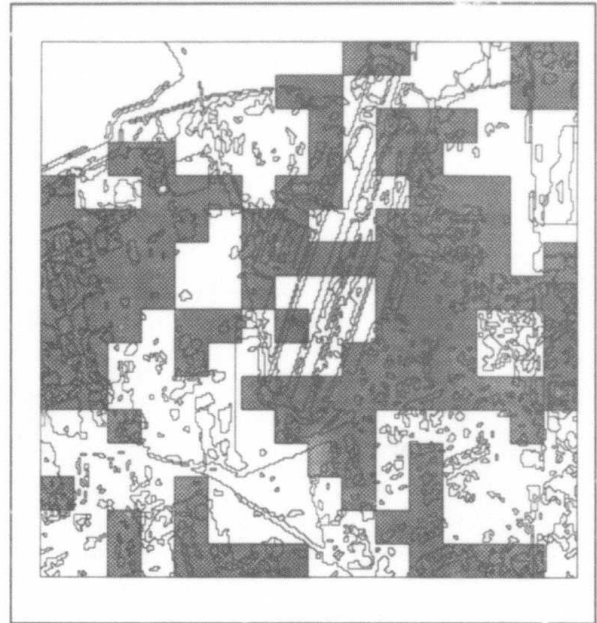


Figure 3.4.

The darkend area shows those sectors which intersect small, bright regions.

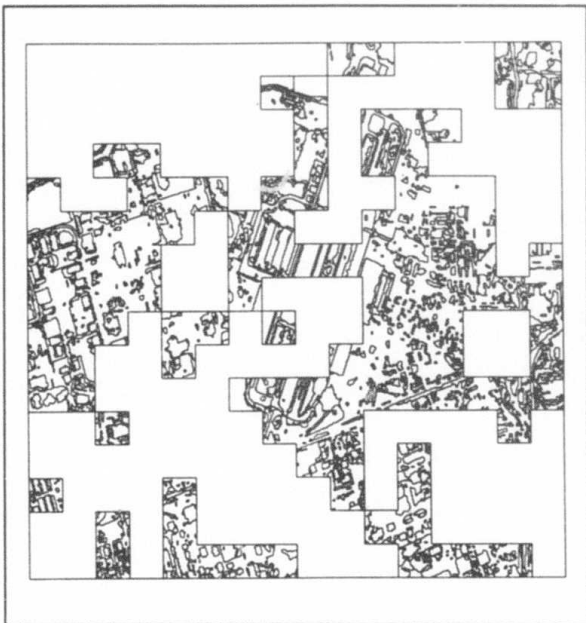


Figure 3.5.

Segmentation at level 9 under the mask created at level 8.

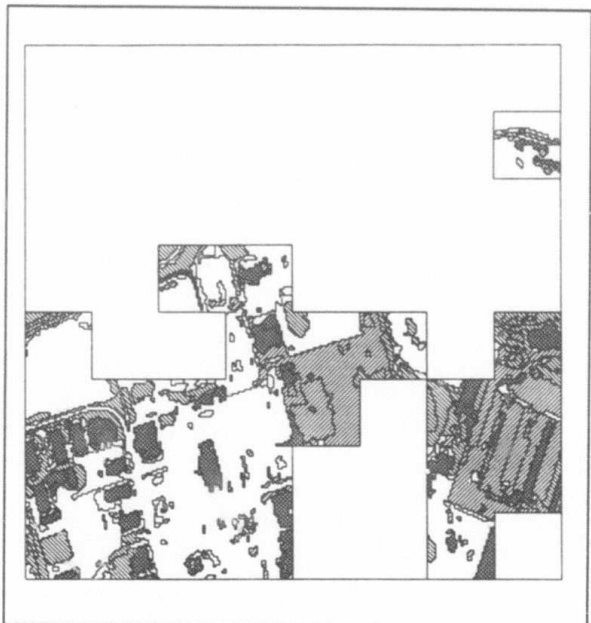


Figure 3.6.

One quadrant of the segmentation at level 9, under the mask generated at level 8. Regions which satisfy a size and intensity constraint are shown as cross-hatched areas.

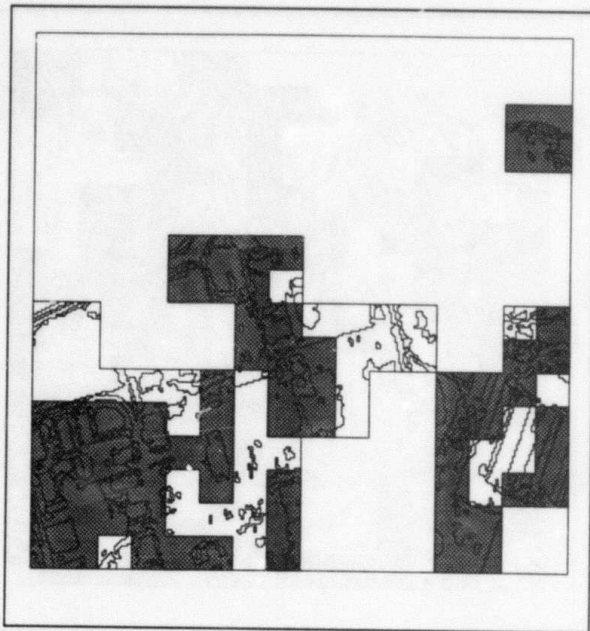


Figure 3.7.

Sectors are selected which intersect regions satisfying a size and intensity constraint



Figure 3.8.

One quadrant of the segmentation at level 10, under the mask generated at level 9. Regions which satisfy a size and intensity constraint are shown as cross-hatched areas.

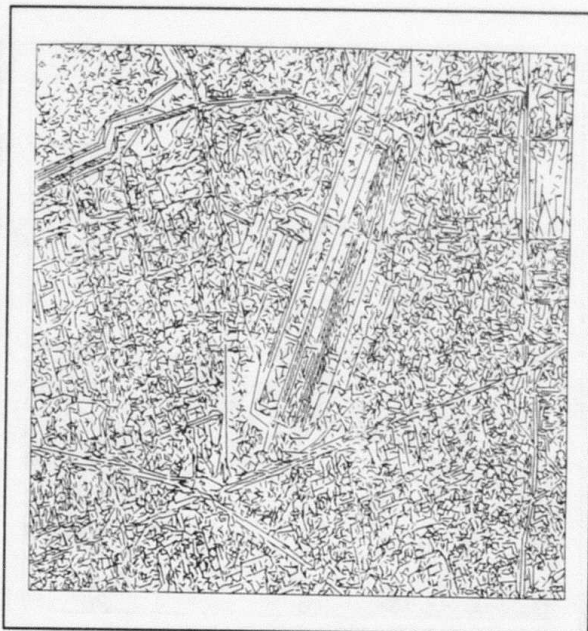


Figure 3.9.

Edges obtained at level 8 from the application of the Burns algorithm. All lines of length greater than 2 pixels are shown.



Figure 3.10.

Edges at level 10 of length greater than 5 pixels and contrast greater than 12.

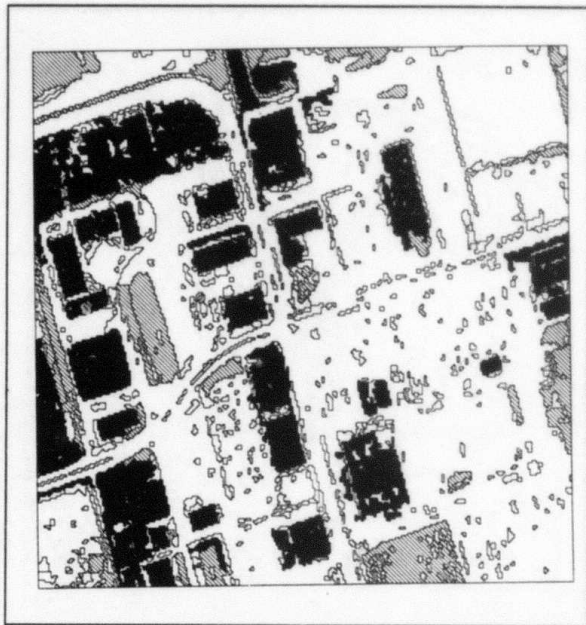


Figure 3.11.

A region is highlighted if the difference between the orientation of the two largest adjacent edge groups is within .15 radian of a right angle.

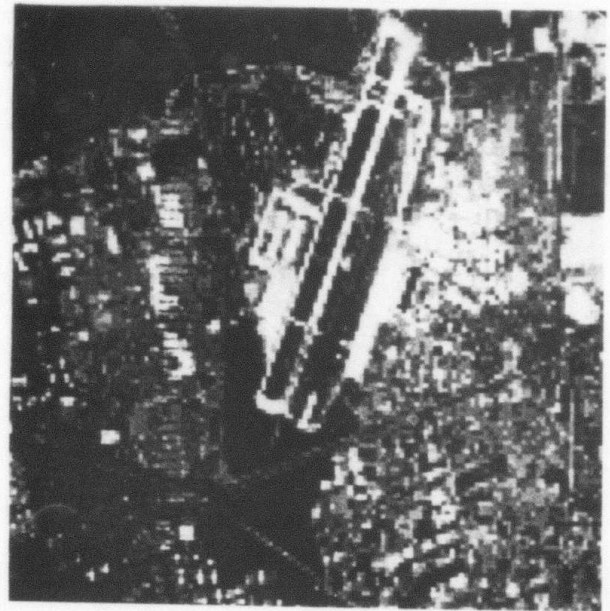


Figure 3.12.

The level 7 image.

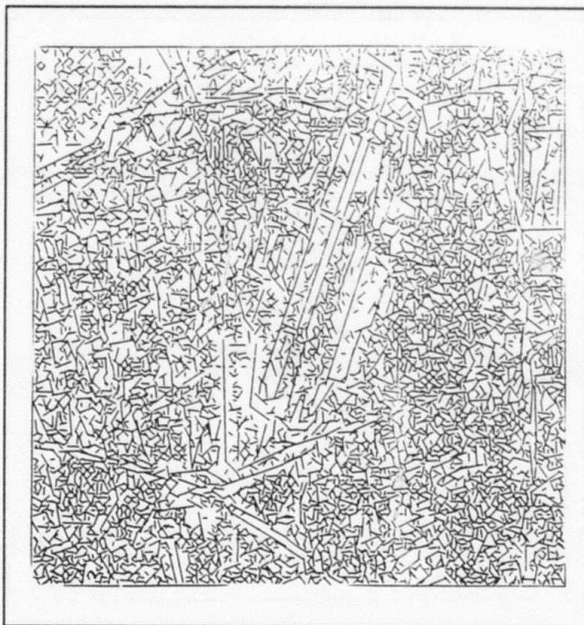


Figure 3.13.

Edges of the level 7 image whose length is greater than 1 pixel.

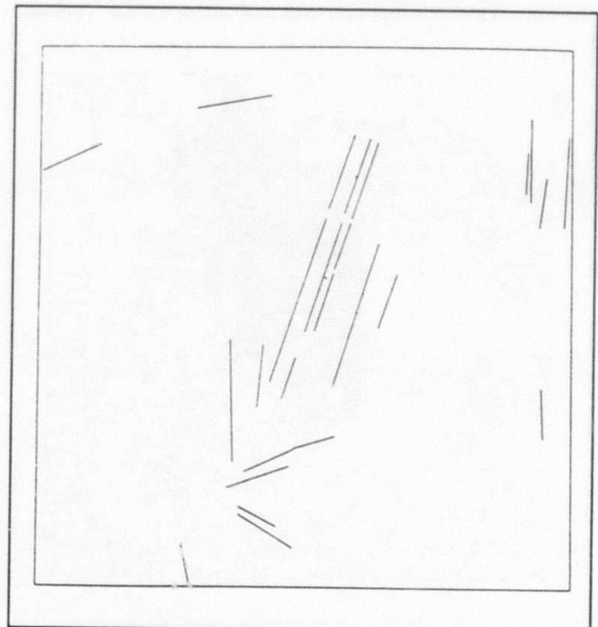


Figure 3.14.

Edges of the level 7 image of length greater than 9 pixels and contrast greater than 12.

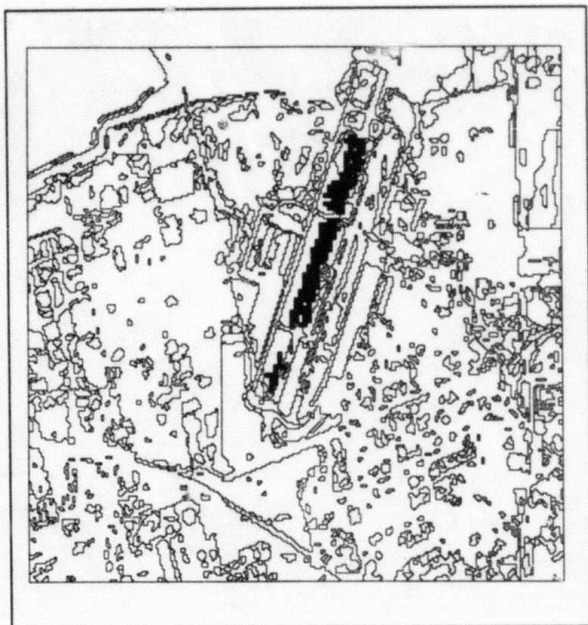


Figure 3.15.

Edge regions of the selected cluster projected from level 7 to level 8, and shown on the level 8 intensity segmentation.

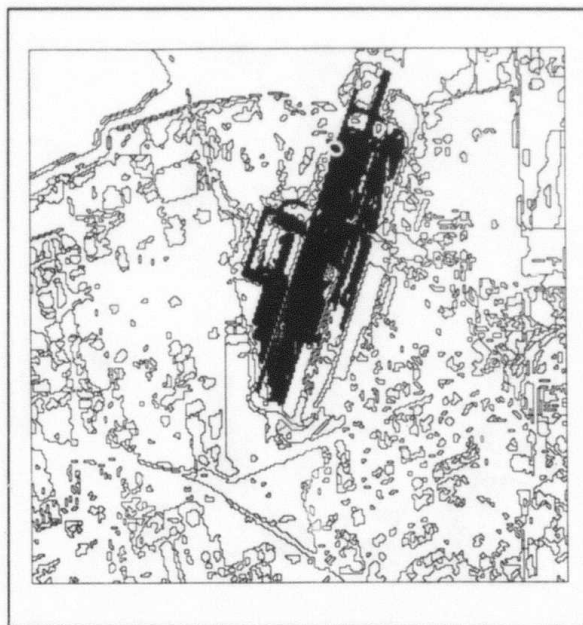


Figure 3.16.

Level 8 intensity regions, adjacent to the projected edge regions.

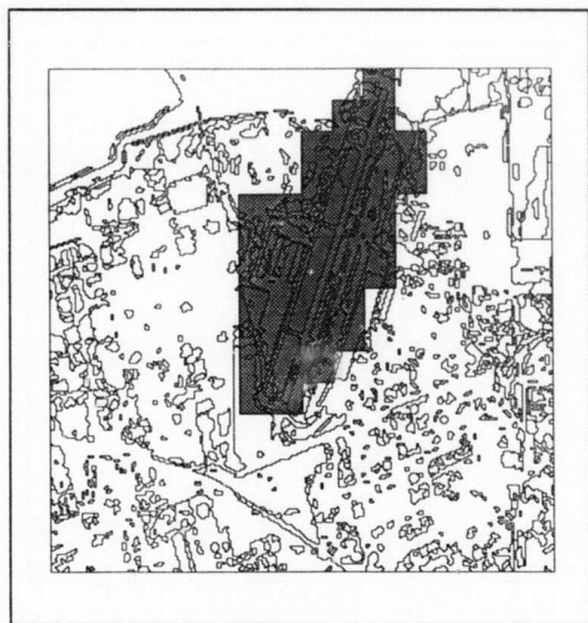


Figure 3.17.

Mask at level 8. A sector is selected if it intersects a region shown in figure 3.16.

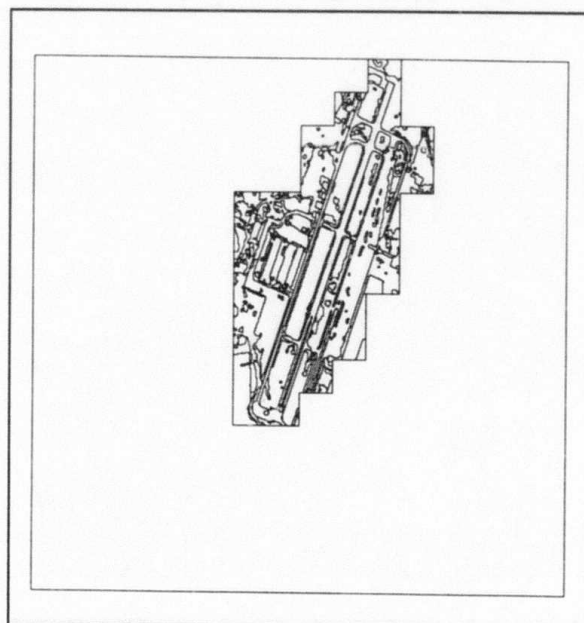


Figure 3.18.

The result of the Nagin-Kohler algorithm at level 9, applied under the mask created at level 8.

AUTOMATIC REFORMULATION OF ALGORITHMS IN COMPUTATIONAL GEOMETRY

Michael Lowry
Artificial Intelligence Laboratory
Stanford University; Stanford CA, 94305

ABSTRACT

Reformulation is a powerful problem solving technique that has received little attention to date. In contrast to previous problem solving techniques, reformulation requires reasoning about a problem formulation rather than within a problem formulation. The research reported in this paper focuses upon two reformulation methods, schema driven reformulation and constraint based reformulation. In schema driven reformulation the original problem formulation is transformed into the terms and vocabulary of a general problem solving method, such as divide and conquer. In contrast, constraint based reformulation focuses upon discovering those properties of a problem that can be exploited by problem solving methods. Schema driven and constraint based reformulation are complementary methods that respectively use the structure and justifications of problem solving methods.

I. INTRODUCTION

This research evolved from a project to automatically choose good representations for problems in vision and robotics. The key insight is that good representations make explicit the problem properties that can be exploited by problem solving methods. For example, choosing a joint centered co-ordinate system as a representation for some robotic problems separates out invariants and simplifies the transformations involved in joint motions. Good vision research is also characterized by finding properties of the image formation process and devising algorithms that exploit these properties. For example, in the blocks world only a small number of the combinatorially possible junction types are physically realizable. This property is exploited in the Waltz algorithm.

Current standard AI problem solving techniques are not capable of solving problems through reformulation. Problem reformulation requires that the initial description be transformed into the terms and vocabulary of an applicable problem solving method. Weak methods, such as generate and test, can often be applied to the problem as stated, whereas strong methods such as dynamic programming can require substantial reformulation. The efficiency of strong methods is derived from their exploitation of problem properties which are only implicit in the initial problem description. The goal of problem reformulation is to discover these implicit problem properties and transform the problem description into terms that make them explicit and operational within the context of a problem solving method.

Saul Amarel has written several papers on problem reformulation including detailed accounts of the steps needed in reformulating the missionaries and cannibals puzzle [Amarel 68] and the tower of hanoi [Amarel 82]. He emphasized the need for discovery of useful problem properties. Ernst and Goldstein have written a special purpose program to transform a problem statement into the parameters for a GPS algorithm through the discovery of operator invariants [Ernst 82]. Similarly, Douglas Smith has worked on the mechanical transformation of problems into Divide and Conquer algorithms [Smith 82]. Korf wrote an article on a general framework for reformulation focusing upon transformation rules that implement homomorphisms and isomorphisms [Korf 80]. He also wrote a special purpose system that transformed a problem statement into the parameters of a specialized extension of GPS called macro problem solving. This requires the discovery of operator decomposability, which is a problem property related to the invariants used in GPS algorithms [Korf 83]. Jack Mostow and Richard Keller have worked on a special type of reformulation called *Operationalization*. The goal of operationalization is to transform a problem statement in a language that cannot be directly executed into the terms of a language that can be directly executed. In their system operationalization is achieved through a sequence of rewrite rules. Mostow has implemented a partially automatic system and demonstrated it on operationalizing advice for the card game hearts [Mostow 83]. Richard Keller presented a paper on finding heuristics for symbolic integration through operationalization [Keller 83].

The focus of the research reported in this paper is reformulation through discovery of exploitable problem constraints and instantiation of general problem solving methods. The first I denote by constraint driven reformulation, the second by schema driven reformulation. The mapping from the original problem description to a reformulated description is a by-product of these two interacting processes. The criterion for evaluating a reformulation is the computational complexity of the resulting algorithm. In contrast to the work cited above, the search for problem properties is not restricted to one particular type of property such as operator invariants, or one particular type of algorithm such as GPS. In addition, the criterion for evaluating a reformulation, which in this work is computational complexity, is used explicitly to constrain the search for useful properties.

The balance of this section gives an overview of schema driven reformulation and constraint based reformulation. Section II describes schema driven reformulation through

an example from computational geometry, and Section III describes constraint driven reformulation. Section IV gives the algorithms for these two reformulation methods.

A schema is an abstract plan for a class of algorithms, examples are :

1. Divide and Conquer
2. Dynamic Programming
3. Iterative Relaxation

A schema specifies the logical and computational relationship between its subparts. The logical relationships between subparts forms the justification of the schema i.e. why it works. When instantiated to a particular algorithm the justification is a proof of program correctness. The computational relationships of control flow and data flow between subparts will be referred to as the structure of a schema.

Schema driven reformulation proceeds by choosing a schema for instantiation and then refining its subparts in order to solve a given problem. For example, after choosing divide and conquer the split step and the merge step would be refined. A well chosen schema can break a difficult problem into more manageable subproblems, i.e. the refinement of the subparts of the schema.

The second general method is to discover the properties of the problem first, and then choose and instantiate a schema based upon this information. The difficulty is in constraining the search space of useful properties. The search space of properties is defined by two lattices in a space of semantic prototypes. A semantic prototype is a set of abstract objects and a set of constraints that hold between the abstract objects. A semi-group is a semantic prototype, in fact most of the semantic prototypes discussed in this paper are algebraic structures. Semantic prototypes have also been referred to as abstractions [Gencsereth 80], semantic clichés [Chapman 83], and the generic term concepts [Lenat 83].

The first lattice in the space of semantic prototypes is formed by the relation *specialization*. A specialization imposes additional constraints on a semantic prototype. For example an equivalence relation is a specialization of a binary relation. The additional constraint is that the relation be reflexive, symmetric, and transitive. The second lattice is formed by the relation *extension*. A semantic prototype is an extension of another if it includes additional abstract objects. For example a monoid is an extension of a semi-group, because it includes an additional element - the identity element. Semantic prototypes can also be linked by mappings, which specify how to transform one semantic prototype into another. For example a mapping specifies how to transform a total order into a sequence.

The search for useful properties is guided by this lattice of semantic prototypes and the criterion of computational complexity. Briefly, the first step is to characterize the objects specified in the problem statement in terms of the semantic prototypes. For example, if a binary relation is part of the problem specification, it is characterized in terms of semantic prototypes such as partial orders, equivalence relations, and total orders. Then algorithms associated with the semantic prototypes are examined to determine whether they meet the criteria specified for computational complexity. If not, then the search for problem properties within the lattice continues until algorithms that meet the specified criteria of computational complexity are found.

Candidate properties are first examined empirically on several examples to determine if they are plausible. For example, the first step in determining whether a binary relation is symmetric is to generate some examples of the relation and then determine whether the examples are symmetric. This is similar to Gelerter's use of models in his geometry theorem prover. If a property is empirically plausible, then an attempt is made to prove that it holds. General purpose theorem provers are too inefficient (e.g. causing a Symbolics 3600 lisp machine to run out of memory). Special purpose theorem provers will be used instead. The details have not been fully worked out, but promising approaches include:

1. Theorem proving schemas such as induction, which are instantiated much like algorithm schemas.
2. Specialized methods for subclasses of theorems. For example, algebraic manipulation is a specialized method for proving algebraic equalities.

II. SCHEMA DRIVEN REFORMULATION

The strategy of schema driven reformulation is to pick a problem solving schema and perform a partial mapping from the problem to the schema. The partial mapping instantiates some of the operators and some of the justifications for steps in the schema. The partially instantiated operators and partially instantiated justifications are then refined through deduction and analysis of examples until a fully instantiated algorithm is derived from the schema.

This strategy will be illustrated with an example from computational geometry:

Given a set of points on the plane, the problem is to find those points that have no other points in their upper right hand quadrant. (See figure 1) More formally, the maximal point problem is formulated as a function from a set of points

S on a plane with an x - y co-ordinate system to a subset T of these points. The condition is that for each member t of the subset T , no other member of S is in the upper right hand quadrant of t . The objective is to find an $N \log N$ algorithm, where N is the size of the set S .

$$\{t \in S \mid (\forall s \in S \{x(t) > x(s) \vee (y(t) > y(s) \vee t = s)\})$$

The initial mapping simply specifies that a set labeled S is the input, that a subset T is the output, and that the input/output assertion specified above holds between S and T . Since the recursive calls are instances of the divide and conquer schema, similar labels and assertions are attached to the input/outputs of the recursive calls. Thus $S1$ is the input of the first recursive call and $T1$ is the output of the first recursive call. Similarly $S2$ is the input of the second recursive call and $T2$ is the output of the second recursive call. The partially instantiated divide and conquer schema is shown in figure 2, the base case and issues of termination will not be considered in order to simplify the example.

Below is a synopsis of the reformulation of this problem into the terms of divide and conquer. Control of the search and branches that ultimately fail will not be considered, though the generation of each move in the reformulation will be elaborated.

The specification of an $N \log N$ algorithm constrains the split and merge steps to be linear time, and the outputs of the split step to be balanced in size. This is an example of how the criteria for judging the merits of a reformulation can constrain the search. The logical constraint on the split step is that it be a partitioning function for the input type, in this case a set. Partitioning functions for a binary split can be implemented using a characteristic function on a set (mapping each element to 1 or 0). A heuristic is the incorporation of a component of the problem specification into the split step; possibly this will appropriately constrain the remaining portions of the schema. One possibility is to specialize the binary relation $x(t) > x(s)$ to a unary function by fixing s . The constraint of balanced outputs requires that s be fixed to the point with median x co-ordinate. This requires a preprocessing step of sorting the points by x -co-ordinate, which fits within the time bounds of $N \log N$.

The next step is to propagate the assertions from this heuristically chosen split step through the schema, in order to constrain subsequent steps. Briefly, the highlights of this propagation are:

1. Disjointness of subsets output from the partitioning
 $\forall s_1 \in S_1, s_2 \in S_2 [x(s_1) < x(s_2)]$
2. T_2 is a subset of T , by definition with respect to the partition S_2 and because of being greater in x co-ordinate for the partition S_1
 $\forall t_2 \in T_2, s_2 \in S_2 [x(s_2) < x(t_2) \vee y(s_2) < y(t_2) \vee t_2 = s_2]$
 $\forall t_2 \in T_2, s_1 \in S_1 [x(s_1) < x(t_2)]$
3. Part of T_1 is a subset of T , namely those points with y co-ordinate greater than y co-ordinate of all points in S_1
 $\forall t_1 \in T_1, s_1 \in S_1 [x(s_1) < x(t_1) \vee y(s_1) < y(t_1) \vee t_1 = s_1]$
 $\forall t_1 \in T_1, s_2 \in S_2 [y(s_2) < y(t_1) \rightarrow x(s_2) < x(t_1) \vee y(s_2) < y(t_1) \vee t_1 = s_2]$

These assertions constrain the merge step to be the union of T_2 and those elements of T_1 which satisfy the third assertion. Unfortunately, the running time for the characteristic function denoted by this assertion is n^2 , which exceeds the linear time bound constraint. Constraint based reformulation, discussed more fully in the next section, determines that assertion 3 can be reformulated in terms of the maximum of S_2 , with respect to the total order induced by y co-ordinate. The cost is now linear for finding the subset of T_1 which satisfies assertion 3. The divide and conquer schema is now fully instantiated as a $N \log N$ algorithm to compute the maximal points.

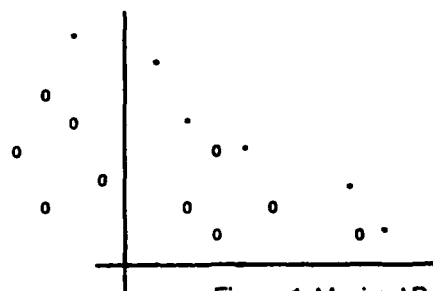


Figure 1. Maximal Point Problem

• Denotes maximal point

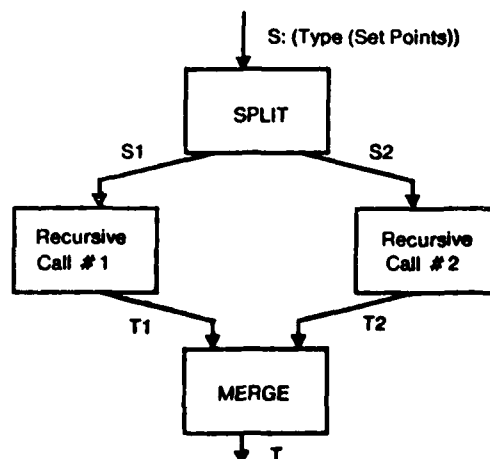


Figure 2. Partially Instantiated Divide and Conquer Schema

III. CONSTRAINT BASED REFORMULATION

Through discovery of implicit constraints and properties inherent in a problem it is often possible to derive representations where the syntax matches the semantics of the problem and the control structure exploits the constraints. Mathematics, particularly abstract algebra, provides a rich vocabulary for describing the structure of many problem domains, in particular the domain of computational geometry.

The first step in constraint based reformulation is to characterize the problem, as given, in terms of basic semantic prototypes. The prototypes often have strong problem solving methods associated with them, reducing the problem of reformulation to one of recognition. More importantly, the recognition of semantic prototypes serves to guide the search for properties that can be exploited by problem solving methods.

Recognition of semantic prototypes proceeds both analytically and empirically for all components of the initial problem description. For the maximal point problem, the generation and examination of several input/output pairs determines that the output is not the null set, and usually not a singleton set. If the output had always been a singleton set, then the problem description could be reformulated as a function from a set of points to a point.

The characteristic function for the maximal points is analysed in terms of its components, which are a binary relation and a quantified variable. The binary relation defined by the disjunction within the characteristic function is analyzed in terms of the basic algebraic properties of binary relations, such as symmetry, transitivity, and functional dependence of arguments. It is found that the relation is neither symmetric nor anti-symmetric, and it is not transitive. Since a relation can be defined in terms of its complement, the complement is also characterized. The complement is found to be antisymmetric and transitive, but not total, and therefore defines a partial order. The problem can then be reformulated as finding the minima of this partial order, or the maxima of its dual.

Definition of partial order and the reformulated problem description:

$$\text{Partial}(s\ t) \Leftrightarrow [x(t) \leq x(s) \wedge y(t) \leq y(s) \wedge t \neq s]$$

$$\{t \in S \mid \forall s \in S \neg \text{Partial}(s\ t)\}$$

Algorithms associated with finding the maxima of a partial order are n^2 , so in this case reformulation through recognition does not improve the computational complexity. However, the recognition of the partial order serves as a guide for exploring possibilities related to partial orders. In particular, total orders are specializations of partial orders whose associated algorithms have very good time complexities. The goal becomes to reformulate the problem description in terms of a total order. In order to be relevant, the total order should be an extension of the partial order, so that if two points are related in the partial order they are necessarily related by the total order:

$$\text{Partial}(s\ t) \rightarrow \text{Total}(s\ t)$$

Two total orders are part of the conjunction defining the partial order, namely ordering by x co-ordinate and ordering by y co-ordinate. Since a conjunction always implies each of its conjuncts, either of these total orders are good candidates. Ordering by x co-ordinate is chosen arbitrarily.

The goal is to recover the maxima of the partial order using the total ordering by x co-ordinate, under the constraint of $N \log N$ time. This goal is split into the subgoal of finding necessary conditions on the maxima given the total order, and the subgoal of finding sufficient conditions on the maxima given the total order. The two subgoals are explored in parallel until one succeeds or both fail. If necessary conditions are found, then the subgoal is set up to find complementary sufficient conditions in order to find an equivalent condition. Similarly, if the subgoal for sufficient conditions succeeds, then the subgoal is set up to find complementary necessary conditions in order to find an equivalent condition.

To find necessary conditions empirically, examples of maxima are generated and relations related to the total order are determined. The empirical approach briefly described below resembles that of Emde [Emde 83]. Random examples of maximal points are generated, and semantic prototypes linked by extension to total orders in the lattice are retrieved, such as minimum, maximum, successor, and monotonicity; which are then used to examine the examples with respect to the ordering by x co-ordinate.

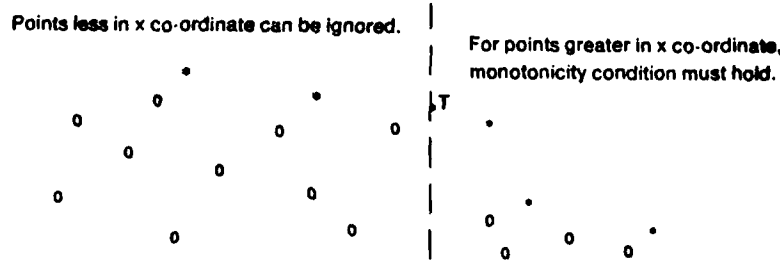


Figure 4. Equivalent condition for a point T to be a maximal point.

The necessary condition found is that the maxima are monotonically increasing in y and decreasing in x . The next step is to find a complementary sufficient condition, in order to develop a reformulated problem description anchored in the ordering by x co-ordinate. This is done by considering sufficient conditions for a point t to be a maxima of the partial order. The ordering by x co-ordinate partitions the rest of the points into those greater than t and those less than t . For those less than t , the first clause in the disjunction ($x(s) < x(t)$) satisfies the maximality condition for t . For those points greater than t , the monotonicity condition is a sufficient and necessary condition - that is, if t is greater in its y co-ordinate than the next maxima in x co-ordinate, then t is also a maxima. These two conditions provide an equivalent condition to the original problem formulation, all that remains is to instantiate a simple enumerate-test-accumulate schema to derive an $N \log N$ algorithm. An $N \log N$ preprocessing step is required to order the points by x co-ordinate. The enumeration is by descending order of x co-ordinate, which implicitly encodes the test for all points less in x co-ordinate. The monotonicity condition instantiates the test operation of the schema. It is constant time because only the previously found maximum needs to be checked. The accumulation is a simple set accumulation.

IV. DESCRIPTION OF REFORMULATION STRATEGIES

A schema serves as an abstract plan for guiding the search for an algorithm. The schema decomposes the reformulation into mutually constraining subparts, defining the relationship between the new terms of the problem description. Schemas are implemented using the formalism of the programmer's apprentice [Rich 81]. Like semantic prototypes, schemas are organized in a dual hierarchy with specialization and extension links. As an example, quicksort is a specialization of Divide and Conquer. An extension to a schema is an addition of new components, such as a preprocessing step to sort the points by x co-ordinate. An algorithm is a schema whose operators and test predicates have been fully specialized to base level operations. The justification for a schema records the logical constraints between subparts and the preconditions for application of a schema.

The strategy of schema driven reformulation is as follows:

1. Choose a kernel schema based upon the known properties of the problem. The preconditions of the schema must not contradict the known properties of the problem. For example, one precondition of divide and conquer is that the input be a partitionable structure.
2. Propagate the known constraints on the problem and the constraints on computational complexity through the schema. Parts of the schema will probably be uninstantiated, unless sufficient constraints on the problem have already been discovered.
3. Choose an uninstantiated component of the schema and instantiate it. Choosing an instantiation for part of a schema is a recursive call on choosing a schema for solving a problem. In this case the uninstantiated component is the new problem. In the process of instantiation, the kernel schema might be extended.
4. Propagate the constraints from instantiating a component through the rest of the schema.
5. If the schema is fully instantiated then terminate, otherwise loop back to step 3.

Constraint based reformulation is essentially goal directed discovery. The objective is to find properties and constraints within a problem that match the preconditions of efficient problem solving methods. It complements steps one and three of the schema driven strategy. In the example of constraint based reformulation, all the problem properties were found before a schema was chosen for instantiation, but in actual operation the strategy is mixed between schema driven reformulation and constraint based reformulation.

Constraint based reformulation is based upon the use of semantic prototypes-abstract structures that capture exploitable regularities in a domain. Semantic prototypes are also organized in a dual hierarchy. For example a group is an extension of a monoid, while a total order is a specialization of a partial order. Often the constraints that define semantic prototypes can be found independently. For example a binary relation is a partial order if it is antisymmetric and transitive. These constraints are found independently and cached, for use in possibly instantiating other semantic prototypes. This substantially reduces the search space. Semantic prototypes are determined both empirically and analytically. The empirical search is done by generating examples and verifying that the examples are consistent with the constraints. The analytic search is done by setting up the goal of proving that the constraints of a prototype hold for some applicable structure such as a binary relation. Special purpose theorem proving techniques are used, since general purpose theorem provers are too inefficient.

The strategy of constraint based reformulation is as follows:

1. Characterize the problem description as stated in terms of semantic prototypes. Although the prototypes will vary between domains, certain prototypes are useful in almost any domain, such as symmetries between arguments to a relation and transitivity. Reformulate the description in terms of the applicable semantic prototypes.
2. The semantic prototypes guide the search for alternative necessary and sufficient conditions. Specializations of a prototype are often associated with efficient problem solving methods. The semantic prototypes are also associated, through extensions, to other potentially useful prototypes. For example monotonicity is an extension of a total order.
3. When a useful necessary condition is found, the subgoal of finding a complementary sufficient condition is pursued. Similarly, when a useful sufficient condition is found, the subgoal of finding a complementary necessary condition is pursued. This usually entails doing a case analysis on the problem, where the partitioning of the problem into cases is based upon the prototypes involved in the necessary or sufficient condition.
4. When an equivalent condition is found to the original problem description, all that remains is to choose and instantiate a problem solving schema.

V. SUMMARY

Problem solving in the domains of vision and robotics typically requires the discovery and exploitation of problem properties only implicit in the initial problem specification. In the future, flexible automatic systems will need to have some ability to automatically reformulate problems. The domain of computational geometry was chosen as a simplified domain for studying methods of reformulation. Two general methods, schema driven reformulation and constraint based reformulation, are described in this paper. A system that uses these two reformulation methods is currently being implemented for the domain of one and two dimensional computational geometry.

ACKNOWLEDGEMENTS

I'd like to thank my advisor, Prof. Binford, for his insights and support. The other members of my thesis committee, Prof. Buchanan and Prof. Lenat, have given useful guidance at critical times. I'd like to thank SRI for the use of their computer facilities, and Dr. Cheeseman and Edwin Pednault for stimulating discussions and advice. This research has been sponsored by DARPA contract N00039-84-C-0211.

Bibliography

- [Amarel 68] Amarel, Saul. "On Representations of Problems of Reasoning about Actions," *Machine Intelligence 3*, 1968 (cited on p. 1)
- [Amarel 82] Amarel, Saul. "Expert Behavior and Problem Representation," *Technical Report CBM-TR-126*, ICSR, Rutgers University, 1982 (cited on p. 1)
- [Chapman 83] Chapman, David. "Naive Problem Solving and Naive Mathematics," MIT-AI Working Paper 249 MIT-AI Lab, Cambridge, MASS, June 1983 (cited on p. 2)
- [Emde 83] Emde, W., C.U. Habel, and C. Rollinger. "The Discovery of the Equator or Concept Driven Learning," *Proceedings IJCAI-83*, Karlsruhe, West Germany August 1983, 455-458. (cited on p. 4)
- [Ernst 82] Ernst, G.W. and M.M. Goldstein. "Mechanical Discovery of Classes of Problem-Solving Strategies," *JACM*, Vol. 29, No. 1 January 1982, 1-23 (cited on p. 1)
- [Genesereth 80] Genesereth, M.R. "Metaphors and Models," *Proceedings AAAI-80*, Stanford, CA, August 1980, 208-211. (cited on p. 2)
- [Keller 83] Keller, R.M. "Learning by Re-Expressing Concepts for Efficient Recognition," *Proceedings AAAI-83*, Washington, D.C. August 1983, 182-186. (cited on p. 1)
- [Korf 80] Korf, Richard. "Towards a Model of Representation Change," *Artificial Intelligence*, Vol. 14 (1980) 41-78 (cited on p. 1)
- [Korf 83] Korf, Richard. "Operator Decomposability: A New Type of Problem Structure," *Proceedings AAAI-83*, Washington, D.C. August 1983, 206-209. (cited on p. 1)
- [Lenat 83] Lenat, D.B. "The Role of Heuristics in Learning by Discovery: Three Case Studies," *Machine Learning, An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto, CA, 1983 (cited on p. 2)
- [Mostow 83] "Machine Transformation of Advice into a Heuristic Search Procedure," *Machine Learning, An Artificial Intelligence Approach*, Tioga Publishing Company, Palo Alto, CA, 1983 (cited on p. 1)
- [Rich 81] Rich, Charles. "Inspection Methods In Programming," MIT-AI-TR-604, MIT-AI Lab, Cambridge, MASS, Ph.D. thesis, June 1981 (cited on p. 5)
- [Smith 82] Smith, D.R. "Top-Down Synthesis of Simple Divide and Conquer Algorithms," *Report NPS52-82-011* Naval Postgraduate School, Monterey, California; November 1982 (cited on p. 1)

A Fast Surface Interpolation Technique.

Grahame B. Smith

Artificial Intelligence Center, SRI International
Menlo Park, California 94025

Abstract

A method for interpolating a surface through 3-D data is presented. The method is computationally efficient and general enough to allow the construction of surfaces with either smooth or rough texture.

1. Introduction

In image analysis we are often faced with the fact that the measurements we make in an image only constrain properties of the 3-D world, instead of specifying them. Analysis techniques that recover 3-D shape information from image measurements incorporate very restrictive assumptions about the nature of the world. In our attempts to avoid the need for these restrictions, we have been examining hypothesis-and-test methods. If we assume that we are able to obtain some shape data, from which we can hypothesize an approximate shape model for the world, then we can use this model to predict image features. To proceed from shape data to an approximate shape model we need to "flesh out" the data. In this paper we address the problem of fitting a surface to a set of points whose 3-D locations are known. While our interest centers on fitting a surface to 3-D location data that have been acquired by processing images of that surface, the technique developed has application to a broad class of surface-fitting tasks.

To select a surface-fitting procedure, it is insufficient merely to know the data set and to require that a surface be fitted to the points in that set. We also need to know the desired properties of the surface, the characteristics of the data, and the uses to which the fitted surface will be put. If we are building a surface to allow, say, water runoff estimates to be made, smoothness may be a desired property for that surface. For realistic rendering of a natural surface in computer graphics, however, a fractal surface may be preferable. While the technique we develop can construct either smooth or rough surfaces, our applications generally require the former. Our examples, Figures 3 and 4, show both types.

Besides the desired properties of the fitted surface, the characteristics of the data limit the approach we must adopt to surface construction. In fitting a surface we must balance the

influence exerted by the data values themselves, against that exerted by the implicit surface model embedded in any fitting procedure. If our data values are inaccurate and we know the class of surfaces that should fit the data, we can usually let the surface model dominate the construction process. Least-square methods are typical of procedures that prefer a model to data. In general, techniques whose resultant surfaces do not conform exactly to the data are known as approximation methods. Methods that produce surfaces conforming exactly to the data are called interpolation methods.

The selection of an approximation or interpolation method is influenced by properties of the data other than their accuracy. Consider, for example, the terrain data collected by a surveyor. In selecting the places at which to make measurements, he considers the breakpoints of the surface - that is, those places on the surface where the gradient is discontinuous - and his data include measurements at these breakpoints. Surface reconstruction by linear interpolation over triangular surface patches is possible because the surveyor has furnished not only the 3-D data, but also an implicit statement that the surface between his points can be approximated by planar patches. In matching stereo pairs of images, an edge-based matcher provides more than the 3-D data it produces. Like a surveyor's data, it too makes an implicit statement about the continuity of the imaged surfaces. On the other hand, an area-based correlation matcher says less about surface continuity, but has the desirable behavior of providing regularly spaced data.

Such data can usually be processed with considerably less computational effort than data that are irregularly spaced. The volume of data, the regularity of their spacing, the implicit characteristics of their collection procedure, and their accuracy are all essential parameters in selecting a surface-fitting technique. For our applications we choose to investigate interpolation methods. We want methods that will work with irregularly spaced data, but still achieve substantial computational savings if we can use a regular grid of data points. We need to be able to handle thousands of such points. As a rule, we do not want to use implicit properties of the data that stem from their collection procedure.

The uses to which the fitted surface will be put further restricts the set of applicable surface-fitting procedures. If the task at hand is surface area estimation, the accuracy of the surface gradients is not important. Conversely, if we wish to use the fitted surface to generate the latter's image under some known lighting conditions, the surface gradient information then becomes crucial. We can classify the uses of fitted surfaces by the surface derivatives that are needed. An application that does

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract MDA903-83-C-0027 and by the National Aeronautics and Space Administration under Contract NASA 9-16064. These contracts are monitored by the U.S. Army Engineer Topographic Laboratory and by the Texas A&M Research Foundation for the Lyndon B. Johnson Space Center.

not require surface derivatives to be calculated can usually be satisfied by a surface composed of local patches. That is, the surface is fitted locally patch by patch, with each patch determined by a small number of local data points. Such methods have strong surface models and few data are needed to instantiate them. As a consequence, however, the surface derivatives are more a function of the surface model than of the data. The amount of data used to determine the surface patch may be barely sufficient to calculate an average value for the surface derivatives across the whole patch; besides, any variations in derivatives across the patch are caused by the model, not the data. The more data employed, the less is the influence of the surface model on the calculation of surface derivatives. In the extreme case, all the data may be used to determine the surface to be fitted at each locality. Such techniques are called global methods, whereas those that use only local data are local methods. Our applications require that we calculate surface curvature from our fitted surface. The technique we present here is a global method for surface fitting.

In summary, we address the problem of fitting a surface to a large data set composed mostly of regularly spaced data points, but which also includes grid points at which we have no data, and non grid points where data values are known. The data are acquired through a collection process that is assumed to yield accurate values, but for which we choose not to characterize the data further. We require a solution that is smooth and from which we can calculate the first and second surface derivatives. We present details of a global interpolation method that is computationally efficient and appears to be applicable to a broad range of tasks. Although the general form of the method applies to non gridded data our computationally efficient algorithm comes from exploiting the regularity of the data points.

We commence by considering the multiquadric method invented by Hardy [1] for modeling natural terrain. In its generalized form, we examine it under the restriction of regularly spaced data points and derive an algorithm to solve for the unknown parameters. We show how to generate the interpolated surface in an efficient manner.

2. Surface Interpolation

2.1 Hyperbolic Multiquadrics

Suppose we have a set of data points, $\{(x_i, y_i, z_i)\}_{i=0}^{n-1}$ in 3-D space to which we wish to fit a hyperbolic multiquadric surface [1] defined by

$$z(x, y) = \sum_{j=0}^{n-1} c_j [d_j^2(x, y) + h]^{\frac{1}{2}},$$

where $d_j^2(x, y) = (x - x_j)^2 + (y - y_j)^2$, h is a user-specified constant, and c_j 's are the coefficients that must be determined.

To understand this method, let us suppose that $h = 0$. The data are fitted by placing a cone at each of the n data points so that the cone's axis is aligned with the z axis direction and

the cone's apex is in the $z = 0$ plane. That is, the data are fitted with a set of cones, some of which are inverted. The z value of the constructed surface at position (x, y) is calculated by summing the z values contributed by each of the n cones at this (x, y) position.

Each cone has one free parameter, namely, its apex angle; we determine these apex angles by requiring that the constructed surface pass exactly through the data points. In the foregoing expression, the c_j 's correspond to the apex angles of the cones. We calculate the c_j 's by solving the $n \times n$ system of linear equations

$$\sum_{j=0}^{n-1} c_j [d_j^2(x_i, y_i) + h]^{\frac{1}{2}} = z_i \quad i = 0, \dots, n-1$$

Note that this fitting technique does not require that the data be regularly spaced; furthermore, when $h \neq 0$, hyperboloids rather than cones are fitted to the data. Cones and hyperboloids are not the only options. Stead [2], for example, has generalized this method, using the form

$$z(x, y) = \sum_{j=0}^{n-1} c_j [d_j^2(x, y) + h]^{\frac{1}{2}}$$

2.2 General Form

We examine surface-fitting techniques that use the general form of the above method, namely,

$$z(x, y) = \sum_{j=0}^{n-1} c_j g(x - x_j, y - y_j)$$

where the kernel function g is any function of the parameters $x - x_j, y - y_j$. Clearly, the previously defined functions are particular cases of this form. As before, we determine the c_j 's by solving the $n \times n$ system of linear equations

$$\sum_{j=0}^{n-1} c_j g(x_i - x_j, y_i - y_j) = z_i \quad i = 0, \dots, n-1$$

For large values of n it is not feasible to solve this system of equations. In our applications n may be 10,000. However, for smaller n we have used the above form to "patch" holes in a regular grid of data points. While any kernel function can be employed, we have found it important to match the method used to solve the $n \times n$ system of linear equations to the form of the kernel function selected. The numerical difficulties encountered in solving some of the systems of equations produced by a particular kernel can often be averted by exploiting properties of the linear system stemming from the choice of kernel function. For example, if we use the Gaussian function as the kernel, the symmetric positive definite coefficient matrix of the system of linear equations allows solution by the "square-root" method (see, for example [3]), and avoids the numerical problems created by Gaussian elimination. If we impose the restriction that the data points must be gridded, we can find feasible solution techniques even when n is of the order of millions.

2.3 Regular Grid Solution

Consider the problem of fitting the surface

$$z(x, y) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} c_{ij} g(x - x_{ij}, y - y_{ij}) \quad (1)$$

where $\{(x_{i,j}, y_{i,j})\}_{i=0, j=0}^{n-1, m-1}$ is an $n \times m$ regular grid, to the data set $\{(x_{i,j}, y_{i,j}, z_{i,j})\}_{i=0, j=0}^{n-1, m-1}$. We can find an expression for calculating the $c_{i,j}$'s in the following manner.

Let $G(u, v)$ denote the discrete Fourier transform (DFT) of $g(x, y)$. Using the shift theorem of DFT theory, we note that the DFT of $g(x - x_{i,j}, y - y_{i,j})$ is $G(u, v)e^{-2\pi i(\frac{x_{i,j}}{n} + \frac{y_{i,j}}{m})}$. If $Z(u, v)$ denotes the DFT of $z(x, y)$, we can write the DFT of Equation (1), namely,

$$Z(u_{k,l}, v_{k,l}) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} c_{i,j} G(u_{k,l}, v_{k,l}) e^{-2\pi i(\frac{x_{i,j}}{n} + \frac{y_{i,j}}{m})} \quad \begin{matrix} k = 0, \dots, n-1 \\ l = 0, \dots, m-1. \end{matrix}$$

Removing $G(u_{k,l}, v_{k,l})$ from the summation, we have

$$\sum_{i=0}^{n-1} \sum_{j=0}^{m-1} c_{i,j} e^{-2\pi i(\frac{x_{i,j}}{n} + \frac{y_{i,j}}{m})} = \frac{Z(u_{k,l}, v_{k,l})}{G(u_{k,l}, v_{k,l})}$$

Taking the inverse DFT of the above expression, we obtain

$$\begin{aligned} \sum_{k=0}^{n-1} \sum_{l=0}^{m-1} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} c_{i,j} e^{-2\pi i(\frac{x_{i,j}}{n} + \frac{y_{i,j}}{m})} e^{2\pi i(\frac{x_{k,l}}{n} + \frac{y_{k,l}}{m})} = \\ \sum_{k=0}^{n-1} \sum_{l=0}^{m-1} \frac{Z(u_{k,l}, v_{k,l})}{G(u_{k,l}, v_{k,l})} e^{2\pi i(\frac{x_{k,l}}{n} + \frac{y_{k,l}}{m})} \end{aligned}$$

Using F^{-1} to represent the inverse DFT, $F^{-1}[\frac{Z(u,v)}{G(u,v)}](x_{p,q}, y_{p,q})$ is the inverse DFT of $\frac{Z(u,v)}{G(u,v)}$, calculated at $(x_{p,q}, y_{p,q})$. We have

$$\begin{aligned} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} c_{i,j} \sum_{k=0}^{n-1} \sum_{l=0}^{m-1} e^{-2\pi i(\frac{x_{i,j}}{n} + \frac{y_{i,j}}{m})} e^{2\pi i(\frac{x_{k,l}}{n} + \frac{y_{k,l}}{m})} = \\ F^{-1}[\frac{Z(u,v)}{G(u,v)}](x_{p,q}, y_{p,q}) \end{aligned}$$

Now if $x_{i,j} = x_{p,q}$, and $y_{i,j} = y_{p,q}$,

$$\sum_{k=0}^{n-1} \sum_{l=0}^{m-1} e^{-2\pi i(\frac{x_{i,j}}{n} + \frac{y_{i,j}}{m})} e^{2\pi i(\frac{x_{k,l}}{n} + \frac{y_{k,l}}{m})} = nm \quad \text{otherwise} = 0$$

Hence

$$c_{i,j} = \frac{1}{nm} F^{-1}[\frac{Z(u,v)}{G(u,v)}](x_{i,j}, y_{i,j}) \quad (2)$$

An alternate way of viewing the above derivation is to note that $g(x_{p,q} - x_{i,j}, y_{p,q} - y_{i,j})$ forms a circulant matrix, and to recall that such matrices are diagonalized by the discrete Fourier transform [4].

2.4 Surface Rendering

Once the $c_{i,j}$'s have been calculated, Equation (1) provides an analytic expression for the constructed surface. We can calculate $z(x, y)$ for any (x, y) position. However, each such calculation involves the sum of nm terms. If it is our intention to use

this analytic expression for surface interpolation we may have to calculate this sum a very large number of times. The cost savings gained in computing the $c_{i,j}$ coefficients by means of the DFT (implemented by the fast Fourier transform) will be offset by the cost of these summations. As a rule, if we commence with data on a regular grid we want an interpolated surface on a finer grid. This results in considerable savings which are realized when the DFT is again employed.

Suppose we want to interpolate each grid interval in x and y at an additional number of points so that the final surface is calculated on a $rn \times sm$ grid. Consider Equation (1), revised for the new, larger grid:

$$z(x, y) = \sum_{i=0}^{rn-1} \sum_{j=0}^{sm-1} c'_{i,j} g(x - x_{i,j}, y - y_{i,j}) \quad (3)$$

where

$$c'_{i,j} = c_{i \div r, j \div s} \quad \begin{matrix} i \pmod{r} = 0, j \pmod{s} = 0, \\ = 0 \quad \text{otherwise.} \end{matrix}$$

That is, we assume the surface is constructed by the placing of objects at each of the new grid points, but zero coefficients are associated with all objects except those placed at the original data points. Now, taking the DFT of Equation (3), we get

$$\begin{aligned} Z(u_{k,l}, v_{k,l}) = \sum_{i=0}^{rn-1} \sum_{j=0}^{sm-1} c'_{i,j} G'(u_{k,l}, v_{k,l}) e^{-2\pi i(\frac{x_{i,j}}{rn} + \frac{y_{i,j}}{sm})} \\ \text{for } k = 0, \dots, rn-1 \\ l = 0, \dots, sm-1, \end{aligned}$$

i.e.,

$$Z(u_{k,l}, v_{k,l}) = G'(u_{k,l}, v_{k,l}) C'(u_{k,l}, v_{k,l})$$

where $G'(u, v)$ is the DFT of $g(x, y)$, defined on the finer grid, and $C'(u, v)$ is the DFT of the array $c'_{i,j}$.

The interpolated surface can be formed by taking the inverse DFT of the above expression:

$$z(x_{i,j}, y_{i,j}) = F^{-1}[G'(u_{k,l}, v_{k,l}) C'(u_{k,l}, v_{k,l})]$$

Note that, in finding the $c_{i,j}$'s in Equation (2), we took the inverse DFT of $\frac{Z(u,v)}{G(u,v)}$, then stretched these $c_{i,j}$'s by adding zeros at the points corresponding to the new interpolation points, and finally took the DFT of the stretched coefficients to calculate $C'(u, v)$. These steps are in fact unnecessary, for we can calculate the $C'(u, v)$'s directly from the $\frac{Z(u,v)}{G(u,v)}$'s. The similarity theorem of DFT theory [5] is required:

$$C'(u_{k,l}, v_{k,l}) = \frac{1}{rs} \frac{Z(u_{k,l \pmod{n}}, v_{k,l \pmod{m}})}{G(u_{k,l \pmod{n}}, v_{k,l \pmod{m}})} \quad \begin{matrix} k = 0, \dots, rn-1 \\ l = 0, \dots, sm-1. \end{matrix} \quad (4)$$

2.5 Algorithm

We can now write down our interpolation algorithm:

1. Given the data array $z(x_{i,j}, y_{i,j})$, find its DFT, $Z(u_{i,j}, v_{i,j})$
2. Find the DFT, $G(u_{i,j}, v_{i,j})$, of the kernel function, $g(x_{i,j}, y_{i,j})$
3. $C(u_{i,j}, v_{i,j}) = \frac{Z(u_{i,j}, v_{i,j})}{G(u_{i,j}, v_{i,j})}$
4. Calculate $C'(u_{i,j}, v_{i,j})$, using Equation (4)
5. Calculate $G'(u_{i,j}, v_{i,j})$ for the larger interpolation grid
6. $Z'(u_{i,j}, v_{i,j}) = C'(u_{i,j}, v_{i,j})G'(u_{i,j}, v_{i,j})$
7. Find the interpolated surface by taking the inverse DFT of $Z'(u_{i,j}, v_{i,j})$.

Note that for selected kernel functions Steps 2 and 5 could be precomputed for standard-size grids. As an alternative, these steps can sometimes be accomplished by analytic means if the analytic form of the kernel function is known.

3. Discussion

For purposes of illustration, let us compare the computational efficiency of this method on a regular grid with the cost of the usual non gridded formulation of the multiquadric. Of course, since the usual formulation deals with irregularly spaced data, we would not expect it to compare favorably with this method; such a comparison nevertheless confirms the advantages of our technique. Consider a square $n \times n$ grid of data points on which we want an interpolated surface over a $rn \times rn$ grid. The usual multiquadric formulation solves a $n^2 \times n^2$ system of linear equation at a cost proportional to n^6 , and calculates $rn \times rn$ sums of terms at a cost proportional to $r^2 n^4$. If it is assumed that $n > r$, this cost is dominated by the n^6 term.

The algorithm outlined above is dominated by the cost of the DFTs in Steps 5 and 7. We use the fast Fourier transform to implement the DFT. This means that we pad our data with zeros to force the dimension size of the grid to be a power of 2. At worst, our grid is $2rn \times 2rn$. The cost of the DFT is proportional to $4r^2 n^2 \log 2rn$. Even if r were as great as n , this cost would be proportional only to $n^4 \log n$. From an empirical standpoint, the algorithm outlined is faster for n (and k) of the order of 10.

The outlined algorithm places little limitation on the type of kernel function employed. Not only smooth, but also rough functions may comprise the basic objects from which the surface is built. We have used, inter alia, cones, hyperboloids, and Gaussian-shaped objects, some of which had fractal texture added to them. In Figures 1-4 we show profile plots. Figure 1 shows a real surface, Figure 2 the sampling grid we used to select data points. In Figure 2 the profiles depict what would have been obtained if we had used bilinear interpolation to build the surface. Figure 3 reveals the resultant surface when Gaussian kernel functions were used, while Figure 4 was obtained with a kernel function that had fractal texture added to a Gaussian base. When we compare the fitted surface to ground truth, the average error for the smooth kernel functions used by us, is approximately one percent of the data height range. As with any fitting technique, we cannot construct surface features that are not described by the sampled data.

We indicated that one reason for investigating global surface interpolation techniques was the need to calculate reliable estimates of surface curvature. In our preliminary trials with

synthetic surface data the constructed surface appears to allow adequate surface curvature estimation. This will be tested further in future applications.

4. Conclusions

We have presented a method of surface interpolation that is computationally efficient. The reconstructed surface is fitted globally to enable the data rather than an implicit surface model to control the construction process. The method makes it possible to build not only the more customary smooth interpolated surface, but the roughly textured type as well.

Surface reconstruction methods provide a means of using the hypothesis-and-test approach in image analysis. They provide a mechanism for using image information that only constrains rather than specifies 3-D world parameters. The outlined algorithm is a tool for hypothesizing a broad range of surface types.

References

1. Hardy, R.L., Multiquadric Equations of Topography and Other Irregular Surfaces, *J. Geophysical Res.*, Vol. 76, No. 8, 1971, pp. 1905-1915.
2. Stead, S.E., Smooth Multistage Multivariate Approximation, Ph.D. thesis, Division of Applied Mathematics, Brown University, Providence, Rhode Island, 1983.
3. Froberg, C-E., *Introduction to Numerical Analysis*, Addison-Wesley Publishing Co., 1969.
4. Hunt, B.R., The Application of Constrained Least Squares Estimation to Image Restoration by Digital Computer, *IEEE Trans. Computers*, Vol. C-22, No. 9, 1973, pp. 805-812.
5. Bracewell, R.N., *The Fourier Transform and Its Applications*, McGraw-Hill Book Co., 1978.

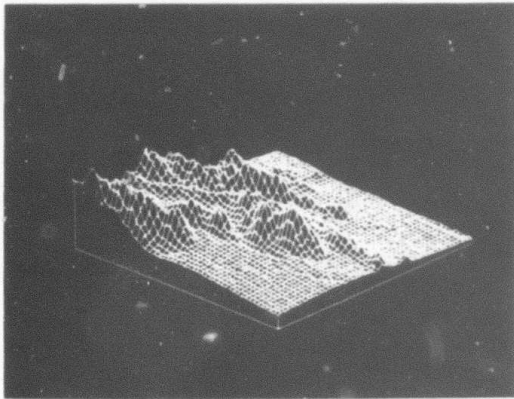


Figure 1. Real Surface Used as Ground Truth

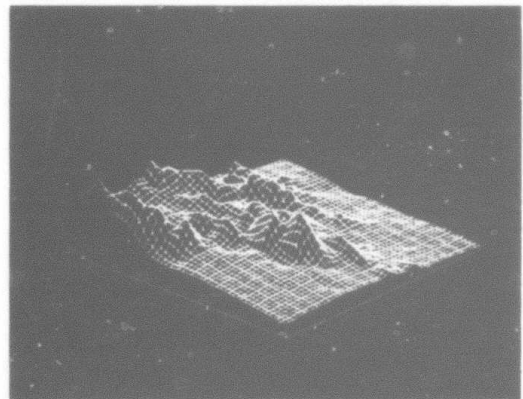


Figure 3. Reconstructed Surface by Means of Gaussian Kernel Functions.

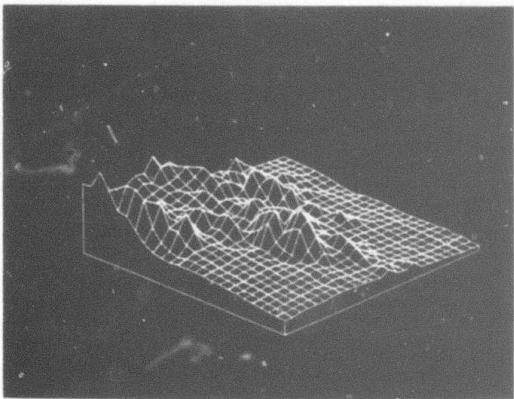


Figure 2. Data Sampled at Grid Intersections

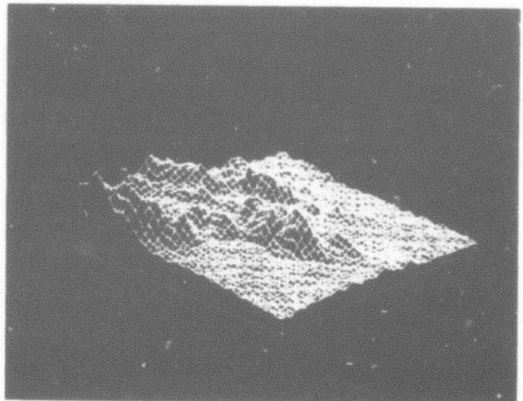


Figure 4. Reconstructed Surface by Means of Fractal Textured Gaussian Kernel Functions.

The Hough Transform Method
on Fine-Grained Tree-Structured SIMD Machines¹

Hussein A. H. Ibrahim
John R. Kender
David Elliot Shaw

Department of Computer Science
Columbia University, New York, NY 10027

Abstract

Recent research at Columbia University has demonstrated that fine-grained tree-structured SIMD architectures, which have favorable characteristics for efficient VLSI implementation, can be used for the rapid execution of a wide range of image understanding tasks. The NON-VON supercomputer, currently being constructed at Columbia University, is an example of such an architecture. In this paper, we describe two algorithms for the implementation of the Hough transform method on NON-VON. The first one may be regarded as a parallel implementation of the standard sequential machine algorithm. The second algorithm treats the NON-VON tree as an independent set of subtrees, resulting in a more efficient algorithm in terms of both execution time and the number of processing elements required. A parallel high level language based on PASCAL is used to describe our algorithms. A description of certain special architectural features of the NON-VON machine that have proven useful for image understanding algorithms is also presented.

1 Introduction

An important goal for researchers in computer vision is to construct vision systems that receive an image or a sequence of images from a sensory device and output an interpretation of this input in real time. Input images with reasonable resolution contain large quantities of data, and conventional von Neumann machines require an excessive amount of time to sequentially process the fetched data. Image understanding applications, however, usually involve computations that can be performed simultaneously on many or all of the image elements. Consequently, parallel computers are highly desirable for fast execution of image understanding tasks.

¹This research was supported in part by the Defense Advanced Research Projects Agency, under contract N00039-84-C-0165, and by the New York State Center for Advanced Technology in Computers and Information Systems at Columbia University.

A computer implementation of a complete vision system requires not only the performance of many computations on large structured arrays of raw image data at the lowest level, but also sophisticated decision making at the highest level. Recent advances in very large scale integrated (VLSI) circuitry have lead to a surge in research aimed at developing new computer organizations that meet the large computational requirements of image analysis tasks. Various kinds of special parallel machines for computer vision have been proposed and some of them have been implemented; examples are described in [Duff 76], [Mark 80], [Pott 83], and [Kush 82].

Recent research at Columbia University has demonstrated that *fine-grained tree-structured single instruction stream multiple data stream* (SIMD) [Flynn 72] computer architectures can be used for the rapid execution of a wide range of vision system tasks. The term "fine-grained machines" refers to machines with a very large number of very small *processing elements* (PE's). A detailed discussion of the efficient VLSI implementation of tree-organized machines can be found in [Ibra 83]. The NON-VON supercomputer, currently being constructed at Columbia University, is a representative example of this class of architectures². Several parallel image understanding algorithms have been developed and implemented on a simulator of the NON-VON machine. The tasks have been selected to span different levels of computer vision. These algorithms incorporate novel approaches to reduce the effects of the communication bottleneck usually associated with tree architectures. Issues related to the implementation of these algorithms, such as image representation and image I/O, are discussed in [Ibra 84b].

In this paper, we describe the implementation on NON-VON of the Hough transform method for

²the first prototype is expected to be completed by late 1984

detecting object boundaries described by parameterized curves. (Other tree-based parallel image understanding algorithms have also been developed for the NON-VON machine. They include the gray-scale image histogram, image correlation, connected component labeling, computing Euler's number for images, set operations, and the computation of the geometric properties of objects.)

In the following section, the architecture of NON-VON is described and compared with other proposed hierarchical architectures for vision. In Section Three, algorithms for the Hough transform method are described.

2 The NON-VON Supercomputer Architecture

The name NON-VON is used to describe a family of massively parallel tree-structured machines intended to support large scale data manipulation [Shaw 84a]. The architectures of all the NON-VON family members include a tree-structured *Primary Processing Subsystem* (PPS) based on custom VLSI circuits, along with a *Secondary Processing Subsystem* (SPS) based on a bank of intelligent disk drives. Figure 1 depicts the top-level organization of the NON-VON architecture.

The PPS is configured as a binary tree of *small processing elements* (SPE's). Each SPE comprises a small RAM (64 bytes in the prototype now under development), a modest amount of processing logic, and an I/O switch. The I/O switch can be set for different modes of communication, as will be described in the next subsection. The SPS is based on a number of rotating storage devices. Associated with each disk head in the SPS is a separate sense amplifier and a small amount of logic capable of dynamically examining the data passing beneath it [Shaw 82]. This organization supports parallel transfer of data between the PPS and SPS, which is necessary to keep I/O from becoming a bottleneck.

NON-VON 1 and NON-VON 3, the first two members of the NON-VON family, include a single special *control processor* (CP) at the root of the tree. The CP is responsible for coordinating different activities within the PPS. It is capable of broadcasting instructions to be executed in all active PE's simultaneously. Thus, NON-VON 1 and NON-VON 3 function for most part as SIMD machines, with all SPE's simultaneously executing the same instruction. Algorithms that use this mode of execution are referred to as SIMD algorithms. NON-VON 3 executes about four million instructions

per second [Shaw 84b]. This number is used throughout the paper to compute the time required to execute the developed algorithms.

The first member of the NON-VON family, NON-VON 1, contains chips with only one PE, and is being constructed primarily to evaluate certain electrical, timing, and layout area characteristics. The chip has already been tested and has been proven functional. A modified version of the chip with eight PE's has been designed for use in NON-VON 3 (a detailed description can be found in [Shaw 84b]). The modified chip has less area per PE, and the instruction set is made more powerful by generalizing register-to-register data transfers and adding more arithmetic processing power. Algorithms described in this paper are based on the NON-VON 3 architecture and instruction set.

In the following subsection, we describe the communication modes in the NON-VON tree, and present a brief description of a PASCAL-based parallel language that is used in describing our algorithms.

2.1 Communication Modes in the NON-VON Tree

Inter-PE communication in NON-VON is supported by the I/O switch, which is a matrix of pass transistors that routes data between the two internal buses and the I/O ports. The NON-VON I/O switch supports three modes of communication:

1. *Global bus communication*, supporting both broadcast by the CP to all PE's in the PPS as required for SIMD execution, and data transfers from a single selected PE to the CP. No concurrency is achieved when data is transferred from one PE to another through the CP using the global communication instructions. An instruction called **RESOLVE** can be used to disable all but a single PE chosen from among a specified set of PE's. This is an example of a hardware *multiple match resolution* scheme, in the terminology of the literature of associative processors. The CP, on executing a **RESOLVE** instruction, is able to determine whether the operation resulted in any PE being enabled or not. The **REPORT** instruction transfers data from the single chosen PE to the CP using global bus communication.

2. *Tree communication*, supporting data transfers among PE's that are physically adjacent within the PPS tree. Instructions support data transfers to the Parent (P), Left Child (LC), and Right Child (RC) PE's. Full concurrency is achieved in this mode, since all nodes can communicate with their physical tree neighbors in parallel.

3. *Linear communication*, in which the whole tree is reconfigured as a linear array of PE's. This mode of communication supports data transfers to the Left Neighbor (LN) or Right Neighbor (RN) PE's in the linear array. Linear communication is useful for applications that require a predefined total ordering of data. Figure 1 shows how the linear logical sequence has been mapped onto the tree structured physical topology of the PPS by inorder enumeration [Knut 73].

The original NON-VON architecture, which was not intended for computer vision applications, differs from other proposed highly parallel hierarchical image understanding architectures (for example, [Tani 83]) in that it does not employ any extra physical links to perform mesh neighbor communication. This is advantageous from a hardware point of view, as it results in a fixed number of pins per integrated circuit chip, independent of the number of PE's on that chip. This makes it possible to increase the size of the tree as chip dimensions scale down by simply embedding more PE's on the chip. Increasing the machine size involves only removing the old PE chips and plugging in the new ones. On the other hand, the lack of mesh connections slows many local operations in which the output value at an image point depends on its own image value and that of neighbor points. Plans for a vision-oriented variant of the NON-VON 3 that includes mesh connections are now being formulated. These hardware modifications with alternative algorithms, will be discussed in [Ibra 84c].

NON-VON's other special hardware features, including its ability to be configured logically as a linear array, its fast global instruction broadcast, and its multiple match resolution scheme, have proven useful in the algorithms we have developed to overcome some of the problems related to the communication bottleneck generally associated with trees.

In the following section, a PASCAL-based parallel language, referred to as NV-PASCAL, is used to describe the NON-VON Hough transform algorithms presented in this paper. The NV-PASCAL contains some features that simplify and make clear the presentation of our NON-VON algorithms. In the remaining part of this section, we describe briefly some features of NV-PASCAL that are relevant to the algorithms described in this paper.

The language has been designed to be used on SIMD architectures. The principal idea behind the design of NV-PASCAL has been to create features that make full use of the machine's parallel capabilities while retaining all of the high level constructs of PASCAL [Baco 82]. For that purpose, one new data type and five extra constructs have been added to standard PASCAL to enable the user to make full use of the parallelism of the machine. Built-in functions that use the NON-VON instruction set allow the programmer to explicitly make use of the NON-VON tree structure. The NV-PASCAL compiler, developed for the NON-VON supercomputer, produces LISP code that is executed on the existing LISP NON-VON Primary Processing Subsystem (PPS) simulator. On the real NON-VON machine, the LISP code generated will actually cause broadcast of the NON-VON machine instructions to the PPS.

Variables whose values are stored in the tree PE's are referred to as *vector variables*, while *scalar variables* refer to those variables stored in the CP. (Vector variables thus represent a vector of values, stored once per PE, while scalar variables are not replicated.) Italics are used to refer to scalar variables, and capital letters to refer to vector variables.

There are two types of statements in NV-PASCAL: sequential and parallel. The sequential statements are those of standard PASCAL, while the parallel statements are those that operate on vector variables. We describe here only the two of these statements that are used in this paper. The assignment statement can be either sequential or parallel. The sequential statement is the assignment statement encountered in standard PASCAL. The parallel assignment statement is one that refers to a variable that is defined as a vector variable. The parallel assignment statement is executed concurrently in all active PE's in the machine.

The WHERE statement is the conditional statement that operates only on vector variables. The form of the WHERE statement is as follows:

```
WHERE <conditional expression>
DO <statement>
[ ELSEWHERE <statement> ] ;
```

It is used to first select only those PE's with vector variables that satisfy the boolean expression. The statement following the DO is then executed only in these PE's. The statement following the optional ELSEWHERE is executed in the subset of the PE's that failed to satisfy the original conditional expression.

An important point to remember is that the WHERE statement generally executes *both* the statement following the DO *and* the statement following the ELSEWHERE (Unless either all or none of the PE's satisfy the condition.)

Built-in functions based on the NON-VON instruction set are used to implement operations that use the tree communication modes of the machine. Function names that start with 'N-' correspond to NON-VON machine instructions, and their parameters correspond to the arguments of these instructions.

3 The Hough Transform on NON-VON

The Hough transform method is used frequently in image understanding tasks to detect the shape of object boundaries described by parametric curves. This method is based on the duality between points on a curve and the parameters of that curve. In his initial work Hough [Hough 62] described a method for detecting straight lines in an image using the slope-intercept parameterization of the line. According to this parameterization, the line equation is expressed as:

$$y = mx + c$$

Suppose that we have a set of image points $\{(x_1, y_1), \dots, (x_n, y_n)\}$ that have a likelihood of being on linear boundaries. In this paper, we refer to these points as *boundary points*. The Hough transform method organizes the boundary points into a set of straight lines as follows. Consider a boundary point (x_i, y_i) in the image plane. The parameters of all lines passing through this point must satisfy the equation:

$$y_i = mx_i + c$$

This equation corresponds to a straight line in the m - c space (*the parameter space*). Thus, the set of

boundary points in the image plane corresponds to a set of lines in the m - c plane. If two boundary points are on a line AB in the image plane with parameters m_1, c_1 , then the two lines corresponding to these two points in the m - c plane would intersect at the point (m_1, c_1) . In fact, all boundary points in the image plane on the same line AB map to lines in the m - c plane that intersect at the point (m_1, c_1) . Thus, the problem of finding the set of lines in the image plane is reduced to that of finding common points of intersection of lines in the parameter space. A better parameterization of a straight line is suggested by Duda [Duda 72], in which the parameters θ - ρ are used, where θ is the angle of the line normal and ρ is the algebraic distance from the origin. The advantage of this parameterization is that the values of θ and ρ are bounded, while in the case of m - c parameterization the values are not bounded. The Hough transform can be extended to detect other curves of analytical parameters [Ball 75], or to detect general curve shapes using edge orientation at the image points and a reference point [Ball 81]. A memory efficient implementation of the Hough transform on sequential machines is described in [Brow 84]. A parallel algorithm based on the Hough transform for detecting a general curve with specific orientation has been developed by Merlin et al [Merl 75].

The implementation of the Hough transform for detecting straight lines on a sequential machine involves a quantization of the parameter plane into a quadrupled grid. The grid size is determined by the acceptable errors in the parameter values, and the quantization is confined to a specific region of the parameter plane determined by the range of parameter values. A two-dimensional array (*the accumulator array*) is then used to represent the parameter plane grid, where each array entry corresponds to a grid cell. For each boundary point, the algorithm on a sequential machine increments the counts in all accumulator array entries that correspond to grid cells along the straight line in the parameter plane. After this step, grid cells corresponding to the accumulator array entries where the count exceeds a certain threshold value are selected as the set of parameters for the image straight lines being sought. The increment of accumulator array counts can be thought of as a process of "voting" by the boundary points for the parameter values of possible curves passing through these points. The time required to execute this algorithm on a sequential machine is proportional to the size s of the grid, plus the number m of boundary points times the number of votes v of each point ($O(s+mv)$). Memory space required is

proportional to the size of the grid.

In what follows, we describe two parallel algorithms to implement the Hough transform on NON-VON. The first one is a direct parallel implementation of the standard sequential algorithm. The disadvantages of this approach are presented, and we describe a second approach that solves these problems. We assume that the boundary points have been detected by some other procedures and that the PE's holding them are marked using a special flag. For the sake of simplicity, we also assume that the curves being sought are straight lines whose equations are expressed using the slope and intercept parameters.

3.1 The NON-VON Hough Transform Algorithm - A Direct Approach

In the sequential machine implementation of the Hough transform, each boundary point casts its votes in the accumulator array by incrementing all the entries corresponding to grid cells along the parameter space curve associated with this point. This process is repeated for all image boundary points. Next, accumulator array entries whose count exceeds a specified threshold value are selected. We now describe how this algorithm is implemented on NON-VON. Each NON-VON PE is associated with a grid cell in the parameter space. This can be performed by a procedure similar to the address initialization procedure described in [Ibra 84a]. In all those PE's, a vector integer variable COUNT is initialized to zero. The coordinates of boundary points are then reported to the CP one point at a time using the RESOLVE instruction. The coordinates of each reported point are then broadcast to all PE's and all those PE's holding a grid cell across the curve in the parameter space corresponding to the reported point increment the vector variable COUNT by one. This step is performed by substituting the broadcast values in the parameter space curve equation and if it satisfies the equation then COUNT is incremented. Each PE whose COUNT variable exceeds the threshold value is marked, and the value of the grid cell associated with it is reported to the CP using the RESOLVE and REPORT instructions. A vector character variable HT is used to flag those boundary points that have not voted yet. The NON-VON PASCAL algorithm that describes the procedure follows:

```

Procedure hough1(thresh: integer);
var
  x, y, m, c: integer;
  vector_var
  COUNT, X, Y: integer;

```

PARAMETER: boolean;

begin

/* 1. Initialize the vector variable COUNT in all PE's. The other vector variables M, C, and HT are assumed to be defined and initialized by the calling procedure. */

```

COUNT := 0;
PARAMETER := 'N';

```

/* 2. Enable all PE's in which the boundary points have not been reported yet. Report the coordinates of a single boundary point using the RESOLVE instruction, and mark this point as reported. If none is enabled then all the boundary points have been reported. In this case start computing the parameter values using the threshold value. */

```

step2:
where HT = 'Y' do N-A1 := 1;
  elsewhere N-A1 := 0;
if N-RESOLVE() = 0 then
  go to step4;
where N-A1 = 1 do
  begin
    HT := 'N';
    N-REPORT8(XADD, x);
    N-REPORT8(YADD, y);
  end;

```

/* 3. Enable all PE's holding the grid cells. Broadcast the reported image point value. Substitute this value in the equation of the parameter space curve. Increment COUNT in all PE's in which the equation is satisfied. Now loop to select another boundary point. */

```

X := x;
Y := y;
if Y = (M * X + C) then
  COUNT := COUNT + 1;
go to step2;

```

/* 4. Broadcast the threshold value thresh. Mark all PE's in which COUNT > thresh. */

```

step4: where COUNT > thresh do
  PARAMETER := 'Y';

```

/* 5. Report the grid cell values held by these enabled PE's one by one using the RESOLVE instruction. */

```

step5:
where PARAMETER = 'Y' do
  N-A1 := 1;
  elsewhere N-A1 := 0;
if N-RESOLVE() = 0 then return();

```


where $N-A1 = 1$ do

begin

PARAMETER := 'N';

N-REPORT8(M, m);

N-REPORT8(C, c);

go to step5;

end;

end

Steps 2 through 4 are executed a number of times equal to the number of boundary points b . Step 5 executes a number of times equal to the number of curves found, which is less than b . Thus, the algorithm takes time proportional to the number of image boundary points $O(b)$. The NON-VON 3 code for this procedure [Ibra 84c] executes about 200 instructions for Steps 2 through 4 (50 μ sec at 4 Mhz). Of these 200 NON-VON 3 instructions, approximately 160 implement the evaluation of the straight line equation. Step 5 executes about 12 NON-VON 3 instructions for each set of parameter values found. Thus, if the image contains 1000 boundary points, the execution time of the algorithm is approximately 53 msec. The number of PE's required by this approach is equal to the number of grid points. If the grid size is larger than the machine size by a factor of k , then the parameter space is divided into k parts. The above procedure is then executed for each of these parts. The time required to execute the algorithm in this case is $O(kb)$.

One disadvantage of this approach is that it requires a NON-VON machine of size comparable to the grid size, despite the fact that many of the PE's will never get their COUNT incremented. Note also that each time a boundary point is broadcast the curve equation has to be evaluated in each PE, which is a time consuming operation as clear from the numbers cited earlier. The second approach we describe below solves these problems. It uses a number of PE's equal to the number of votes cast by the boundary points, rather than the grid size, and the curve equation is evaluated only once.

3.2 The NON-VON Hough Transform Algorithm - A MSIMD Approach

In our improved approach, the NON-VON tree is treated as if it were an independent set of subtrees, and each boundary point casts its votes one by one in one of these subtrees. This voting process is performed concurrently in all the subtrees. Thus, in time proportional to the number of votes cast by each boundary point, all votes are cast and stored throughout the tree. The problem of finding the parameter values which exceed the threshold value is

equivalent to that of finding the local peaks of a two-dimensional histogram. Because of the way the votes are cast in this second approach, we refer to this algorithm as a multiple-SIMD (MSIMD) algorithm.

The size of these subtrees is determined by the number of votes cast by each boundary point. For example, if each boundary point casts 60 votes, then the subtree size required is at least 60 (A subtree of six levels will suffice). Boundary points are stored in the roots of these subtrees. This can be performed by more than one method. The simplest one is to report the boundary points to the CP one by one using the RESOLVE instruction, and then to broadcast them to be stored in the roots of the subtrees. There are other methods to perform this process more efficiently, but they are beyond the scope of this paper.

The PE's in these subtrees are enumerated in such a way that each PE in a subtree is assigned a unique address (stored in the local variable ADDRESS) in the range $[0, \text{max_num_votes}]$, where max_num_votes is the value of the maximum number of votes casted by each point. The enumeration is performed in such a way that all PE's in the same relative position within these subtrees have the same address. This enumeration procedure is very simple, and will not be described in this paper. Time required by this procedure is proportional to the height of the subtree. We now describe the algorithm for storing the votes in the NON-NON tree.

We assume that the boundary points reside in the roots of the subtrees, with the PE's being enumerated as described before. We also assume that the parameter space is a two-dimensional one. The vector integer variables X and Y are used to store the value of the boundary points, while the vector variables M1 and M2 are used to store the parameter values. A scalar variable g_m1 stores the value of parameter M1 to be broadcast, and the scalar constant delta_m1 is the increment used to change the value of g_m1 . The scalar constant $h_subtree$ is the height of the subtree. The NV-PASCAL voting procedure follows:

Procedure *hough2*,

var

i, j, g_m1 : integer;

vector_var

M1, M2, X, Y: integer;

begin

/* 1. Initialize the scalar variables. The scalar variables $h_subtree$, delta_m1 , max_num_votes are initialized by the calling procedure. */


```

i := 0;
g_m1 := 0;

/* 2. Enable all PE's that are not the root of some
voting subtree. The variable SUBTREE_ROOT is
assumed to be set by the calling procedure. Set X
and Y in each child equal to X and Y in its parent.
Repeat this step h_subtree times. */

where SUBTREE_ROOT = 'N' do
begin
  N-RECV8(P, XADD, X);
  N-RECV8(P, YADD, Y);
  for j = 1 to h_subtree-1 do
  begin
    N-RECV8(P, X, X);
    N-RECV8(P, Y, Y);
  end;
end;

/* 3. Enable the PE's with ADDRESS equal to i in
voting subtrees. In all the enabled PE's store the
new value of the parameter M1. */

step3:
where ADDRESS = i do
  M1 := g_m1;

/* 4. Increment g_m1 by delta_m1, and increment i
by 1. If all the values of M1 have been stored in the
voting PE's, then proceed to compute the value of
M2 in those PE's; otherwise repeat step 3. */

i := i + 1;
g_m1 := g_m1 + delta_m1;
If i < max_num_votes then
  go to step3;

/* 5. Enable all PE's. Using the values of X, Y, and
M1, compute M2 such that the curve equation is
satisfied. */

M2 := compute_m2(X, Y, M1);
end.

```

Step 2 is executed a number of times equal to the subtree height ($\log v$), where v is equal to the number of votes cast by each point. Steps 3 and 4 are executed a number of times equal to v . Thus, the procedure to store the votes in the subtree takes time of $O(v)$. Note that step 5, the evaluation of the curve equation, is executed only one time. If the evaluation of the curve equation results in more than one M2 value for each value of M1, then each PE stores more than one parameter set values. This case depends on the parameter space curve, and should result in a slightly modified version of the algorithm to compute the local peaks of the parameter histogram described later.

The NON-VON 3 code for this procedure executes approximately $(10v + 160)$ NON-VON 3 instructions. For v equal to 100, the time required to cast the votes in the tree is thus about 0.3 msec. If there are more votes than the NON-VON tree size, each PE stores more than one vote. In this case, if each PE stores k values, then the time required to execute the above procedure is $O(kv)$, where k is the ratio between the total number of votes and the NON-VON tree size.

Next, we describe how to find the parameter values that have votes exceeding the threshold value. These values occur at the the local peaks of the two dimensional histogram of the votes for M1 and M2. We assume in the following discussion that there are few of these local peaks. This is a realistic assumption, as the number of curves being sought is usually small. Figure 2 shows such a histogram. In this example, there are a few areas of voting activity (local peaks). A direct approach to the identification of these local peaks involves dividing the two dimensional histogram space into grid cells. For each grid cell, all PE's with M1 and M2 values falling within this grid cell are then marked and counted. The time required to execute this simple procedure is $O(sh)$, where s is the grid size and h is the NON-VON tree height. Counts that exceed the threshold value are the parameter values being sought. A large percentage of the time in this procedure is spent counting votes in grid cells corresponding to areas that contain few votes.

We are currently simulating a different approach, in which areas of non voting activity are not considered in locating the local peaks of the two-dimensional histogram. The procedure first computes a one-dimensional histogram of the parameter M2, as shown in Figure 2. (A pipelined-SIMD algorithm to compute the one-dimensional histogram is described in [Ibra 84c].) A small number of local peaks corresponding to regions of the two-dimensional histogram where most of the votes occur, appear in the one-dimensional histogram. Only votes in those regions are then marked. A second one-dimensional histogram of the parameter M1 is then computed for the marked votes only. The local peaks of this histogram are the values of M1, for which there are local peaks in the two-dimensional histogram. The values of M1 and M2 for which exist local peaks of the two one-dimensional histograms mark the regions of activity in the two-dimensional histogram. These regions of active voting are then checked for exact vote counts. Round off errors in computing the

parameter values can result in peaks that are relatively flat. For this reason, a small window around the regions of voting activity should also be checked when counting the exact votes. This second approach executes in time of $O(m1 + m2 + h)$, where $m1$ and $m2$ are the number of bins in the two one-dimensional histograms. The computation of a 64 bins one-dimensional histogram requires about one msec. The algorithm for locating the local peaks in the two-dimensional histogram of the parameter values as described earlier executes in about 5 msec. The total execution time of the second approach is thus about 5.3 msec, which is considerably less than the time required by the first approach (50 msec for 1000 boundary points).

The algorithms described here can be extended using slight modifications to deal with parameter spaces of higher dimensions. For example, in the first approach if we have an n -dimensional parameter space, then each PE will correspond to a n -dimensional grid cell in this space. In the second approach, the subtree size will correspond to that of $(n-1)$ -dimensional area of the parameter space, and each PE will store parameter values that represent cells in this sub-parameter space. A second approach to extend the Hough transform to parameter spaces of higher dimensions involves applying the current algorithms to two-dimensional cross sections of the multi-dimensional parameter space.

4 Conclusion

In this paper, we have addressed the problem of implementing the Hough transform method on fine-grained tree-structured SIMD machines. Two algorithms have been developed for implementing the Hough transform method on the NON-VON machine. The first one is a parallel implementation of the standard sequential machine algorithm. The second algorithm incorporates novel approaches to exploit the tree organization of the machine, and it executes 10 times faster than the first algorithm.

Bibliography

- [1] Bacon, D., Ibrahim, H., Newman, R., Piol, A., and Sharma, S. "The NON-VON PASCAL." Columbia University, May, 1982.
- [2] Ballard, D. H. "Generalizing the Hough Transform to Detect Arbitrary Shapes." *Pattern Recognition* 13, 2 (1981), 111-122.
- [3] Ballard, D. H., and Brown, C. M. *Computer Vision*. Prentice Hall, 1982.
- [4] Brown, C. M. "Peak Finding with Limited Hierarchical Memory." Proceedings of the 7th International Conference on Pattern Recognition, Montreal, 1984.
- [5] Duda, R. O., Hart, P. E. "Use of the Hough Transformation To Detect Lines and Curves in Pictures." *Communications of the ACM* 15, 1 (January 1972).
- [6] Duff, M. J. B. "A Large Scale Integrated Circuit Array Parallel Processor." Proceedings of the IEEE Conference on Pattern Recognition and Image Processing, 1976, pp. 728-733.
- [7] Flynn, M. J. "Some Computer Organizations and Their Effectiveness." *IEEE Transactions on Computers* 21, 9 (September 1972).
- [8] Hough, P. V. C. "Methods and Means to Recognize Complex Patterns." *US. Patent 3069654* (1962).
- [9] Ibrahim, H. A. H. "Tree Machines Architecture and Algorithms." Columbia University, June, 1983.
- [10] Ibrahim, H. A. H. "The Connected Component Algorithm on the NON-VON Supercomputer." Proceedings of the IEEE Computer Society Workshop on Computer Vision Representation and Control, 1984, pp. 37-45.
- [11] Ibrahim, H. A. H. "Some Image Understanding Algorithms on Fine-Grained Tree-Structured SIMD Machines." Proceedings of the Workshop on Algorithm-Guided Parallel Architectures for Automatic Target Recognition, 1984.
- [12] Ibrahim, H. A. H. *Image Understanding Algorithms on Fine-Grained Tree-Structured SIMD Machine*. Ph.D. Th., Columbia University (in preparation), 1984.
- [13] Knuth, D. E. *The Art of Computer Programming*. Addison Wesley, 1973.
- [14] Kushner, T., Wu, A. U., and Rosenfeld, A. "Image Processing on ZMOB." *IEEE Transactions on Computers* 31, 10 (October 1982).
- [15] Marks, P. "Low Level Vision Using an Array Processor." *Computer Graphics and Image Processing* 14 (1980), 281-292.
- [16] Merlin, P. M., and Farber, D. J. "A Parallel Mechanism for Detecting Curves in Pictures." *IEEE Transactions on Computers* 24, 1 (January 1975).
- [17] Potter, J. L. "Image Processing on the Massively Parallel Processor." *IEEE Computer Magazine* 16, 1 (January 1983).

[18] Shaw, D E The NON-VON Supercomputer.
Columbia University, August, 1982

[19] Shaw, D E SIMD and MSIMD Variants of the
NON-VON Supercomputer. Proceedings of the
COMPCON Spring '84, February, 1984

[20] Shaw, D E, and Sabety T. M. An Eight-
Processor Chip for a Massively Parallel Machine.
Columbia University, July, 1984

[21] Tanimoto, S L A Pyramidal Approach to
Parallel Processing University of Washington,
January, 1983

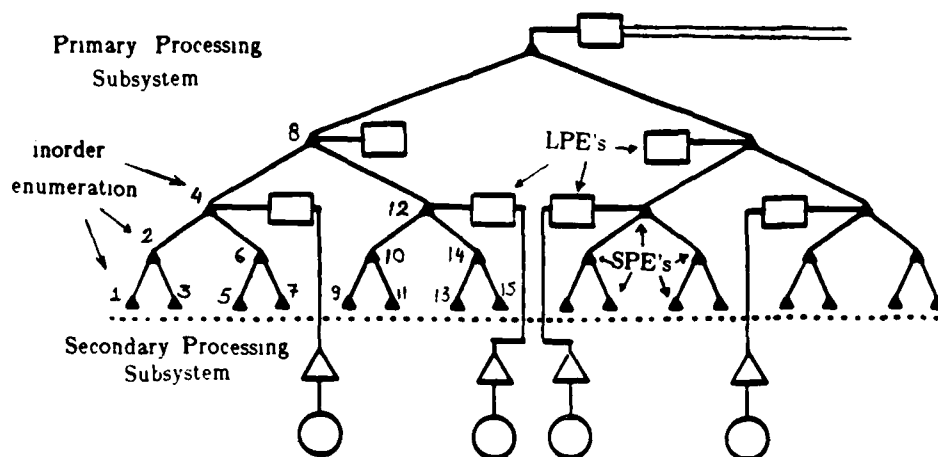


Figure 1: Top-Level Organization of
NON-VON

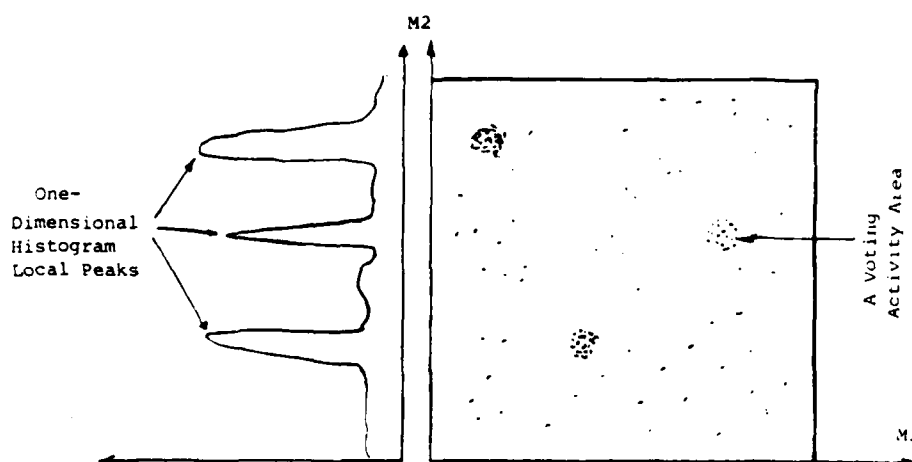


Figure 2: The Two-Dimensional
Histogram of Parameter Values

Controlled-Smoothness Stabilizers for the Regularization of Ill-Posed Visual Problems Involving Discontinuities

Demetri Terzopoulos
MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

Abstract

Through regularization analysis, ill-posed visual reconstruction problems can be restated as well-posed variational principles whose stabilizing functionals impose smoothness constraints on possible solutions. Encounters with discontinuities present serious difficulties to standard regularization methods, however, since the smoothness properties of stabilizing functionals must be regulated appropriately near known singularities. After examining the close connections between regularization and generalized spline approximation, we propose a class of controlled-smoothness stabilizers which overcome potential difficulties with discontinuities.

1. Introduction

1.1. Reconstruction, Splines, & Regularization

Early vision can be described as the reconstruction of physically-based intrinsic scene characteristics from images [Barrow and Tenenbaum, 1978; Marr, 1982]. Included among early visual reconstruction problems are the recovery of lightness from image irradiance, motion fields from time-varying images, and disparity fields from binocular images. An intermediate reconstruction problem concerns the integration of this information with constraints derived from shading, texture, and contours to compute visible-surface representations [Terzopoulos, 1984].

Extending the theoretical aspects of our work on surface reconstruction, we have been able to unify a broad range of visual reconstruction problems in terms of well-posed (quadratic) variational principles in Sobolev spaces (see [Terzopoulos, 1982] section 6). The variational principles define constrained spline approximation problems which involve a particular class of generalized multidimensional spline functionals whose properties have been studied extensively by Duchon [1976, 1977], Meinguet [1979], and others.

As is generally the case for inverse mathematical problems, visual reconstruction problems tend to be ill-posed, in that existence, uniqueness, and stability of solutions cannot be guaranteed in the absence of additional constraints. This important observation was made by Poggio and Torre [1984], who further point out that systematic approaches to the solution of ill-posed problems can be exploited in early vision. They investigate in this context the regularization methods introduced by Tikhonov [1963] and others (see [Tikhonov and Arsenin, 1977] and references therein).

Through regularization analysis, ill-posed visual problems can be restated as well-posed variational principles by the introduction of appropriate stabilizing functionals, notably the stabilizers suggested by Tikhonov [Tikhonov and Arsenin, 1977, pp. 69-70]. Tikhonov's stabilizers can be viewed as spline functionals that impose smoothness constraints on the admissible solutions (typically restricting them to Sobolev spaces of smooth functions). Pragmatically then, generalized spline approximation turns out to be essentially equivalent to

regularization analysis. By exploring this relationship in some detail, our earlier work can be shown to be fully compatible with Poggio and Torre's promising new regularization framework for early vision (also [Poggio, et al., 1984]).

1.2. Discontinuities & Regularization

Smoothness constraints are applicable to the regularization of visual reconstruction problems inasmuch as the coherence of matter tends to produce smooth surfaces relative to the viewing distance over some range of scales. An inevitable complication arises, however, due to the necessity of dealing with discontinuities in the intrinsic properties of scenes. Discontinuities result from significant, spatially localized physical changes in the world, such as abrupt changes in surface geometry (e.g. occlusions), abrupt alterations of surface composition (e.g. texture), or abrupt transitions of illumination (e.g. shadows). Discontinuities tend to be spatially organized along contours in the image, especially when they are due to surface geometry changes. Some discontinuity contours can persist across all scales.

Smoothness assumptions clearly do not hold indiscriminately across discontinuity contours. In this paper, we propose a way to manipulate the smoothness properties of multidimensional Tikhonov stabilizers over the spatial domain. We show that smoothness can be regulated locally to preserve discontinuities, through the specification of a set of parametric functions provided by the stabilizer.

1.3. Ill-Posed Nature of Visible-Surface Reconstruction

The basic ideas derive from a formal treatment of surface discontinuities in the context of visible-surface reconstruction [Terzopoulos, 1983b, 1984]. The task is to reconstruct dense representations of the shapes of visible surfaces from initial measurements of surface depth and orientation (as well as their discontinuities). Such initial measurements are provided by numerous specialized low-level visual processes.

Let the true distance at each point (x, y) in the image from the viewer to visible surfaces be given by the function $Z(x, y)$. This function is often referred to as the depth map of the scene. In general, low-level visual modules generate initial data of the form

$$d_i = L_i[Z(x, y)] + \epsilon_i,$$

where L_i denote measurement functionals of $Z(x, y)$ and ϵ_i denote associated measurement errors. The simplest measurement functionals are evaluation functionals, which provide local depth data of the form

$$L_0[Z(x, y)] = Z(x_i, y_i) = z_{x_i, y_i},$$

and derivative functionals, which provide the local gradient data

$$L_1[Z(x, y)] = Z_x(x_i, y_i) = p_{x_i, y_i},$$

$$L_2[Z(x, y)] = Z_y(x_i, y_i) = q_{x_i, y_i},$$

hence, the local surface normal $n(x, y) = [Z_x(x, y), Z_y(x, y), -1]$. Other functionals such as directional derivatives may also play a role, and they can be accommodated straightforwardly.

Visible-surface reconstruction is a mathematically ill-posed visual

problem, as suggested (informally) by the following considerations: First, the initial measurements are contributed not by one, but by multiple specialized low-level visual processes. Hence, slightly inconsistent measurements (provided by different low-level processes) that happen to coincide will locally overdetermine surface shape. Second, the measurements are not dense, but scattered sparsely over the visual field. Thus, while they constrain surface shape locally, they do not determine it uniquely everywhere. Third, the measurements are not precise, but subject to errors and noise. Indeed, additive noise of high enough frequency, regardless how small its RMS amplitude, can locally perturb surface orientation radically.

Visible-surface reconstruction is thus a fundamentally ill-posed problem, since according to the above three considerations (respectively) we cannot conclude in general that the solution will exist, nor that it will be unique, nor that it will be stable with respect to perturbations in the data.

Given its ill-posed nature, the this visual problem can be approached via regularization methods. In this context, the *thin plate surface under tension*, introduced in [Terzopoulos, 1984] as a physical model for the reconstructed surface, can be readily interpreted as a *controlled-smoothness stabilizer*. We will explain shortly how this special case of generalized controlled-smoothness splines enables the surface reconstruction problem to be regularized, even in the presence of discontinuities.

2. Optimal Approximation with Generalized Splines

The abstract theory of optimal spline approximation is well-developed and a close connection has been established with variational principles involving the constrained minimization of semi-norms in Hilbert function spaces [Laurent, 1972]. This connection leads to natural multidimensional generalizations of the classical univariate splines [Ahlberg et al., 1967] (not as tensor products, but rather through physical interpretations concerning equilibria of elastic bodies). In this section, we consider a class of generalized spline approximation problems and indicate their close relationship to regularization analysis.

Let \mathcal{H} be a linear space of smooth functions and let S be a functional defined on \mathcal{H} which measures the smoothness of a function in \mathcal{H} . Furthermore, let C be a functional on \mathcal{H} which provides a measure of the incompatibility between the function and given data. Consider the following variational principle:

VP: Find $u \in \mathcal{H}$ such that

$$\mathcal{E}(u) = \inf_{v \in \mathcal{H}} \mathcal{E}(v),$$

where

$$\mathcal{E}(v) = S(v) + C(v).$$

This defines an approximation problem — to find the smoothest admissible function in the space \mathcal{H} which is most compatible with the data.

Assuming independently distributed measurement errors ϵ_i with zero means and variances σ_i^2 , the natural incompatibility measure is a weighted Euclidean norm of the discrepancy between the admissible function and the data d_i . This can be written as

$$C(v) = \sum_i \omega_i (I_i[v] - d_i)^2,$$

where the ω_i are nonnegative real-valued weights. The basic measurement functionals of immediate concern are generalized derivative functionals of the form $I_k[Z] = \frac{\partial^k Z}{\partial x_1^{k_1} \dots \partial x_d^{k_d}} \Big|_{\mathbf{x}_i}$. Note, that for $k=0$, I reduces to the evaluation functional $I[Z] = Z(\mathbf{x}_i)$.

2.1. Generalized Spline Functionals

Duchon [1976, 1977] and Meinguet [1979] study the following class of generalized smoothness functionals, defined for d -dimensional functions $v(\mathbf{x})$, $\mathbf{x} = [x_1, \dots, x_d]$:

$$\begin{aligned} S_m(v) = |v|_m^2 &= \sum_{i_1, \dots, i_m=1}^d \int_{\mathbb{R}^d} \left(\frac{\partial^m v(\mathbf{x})}{\partial x_{i_1} \dots \partial x_{i_m}} \right)^2 d\mathbf{x} \\ &= \int_{\mathbb{R}^d} \sum_{j_1, \dots, j_d=m} \frac{m!}{j_1! \dots j_d!} \left(\frac{\partial^m v(\mathbf{x})}{\partial x_1^{j_1} \dots \partial x_d^{j_d}} \right)^2 d\mathbf{x}. \end{aligned}$$

These functionals have several interesting properties [Meinguet, 1979]. Since the variational principle reduces in the one-dimensional case ($d=1$) to the approximation problem associated with classical smoothing splines [Schoenberg, 1964; Reinsch, 1967], the functionals $S_m(v)$ can be thought of as generating multivariate generalized splines. The positive integer m dictates the order of the partial derivatives that occur in the functional, which in turn determines the order of continuity that the admissible functions v must possess.

In fact, the functionals define the natural semi-norms of Beppo-Levi spaces (which are related to Sobolev spaces). These semi-norms are invariant under translation, rotation, and similarity transformation, invariance properties which prove essential in the context of visual reconstruction problems (since the solutions to visual reconstruction problems should not change shape when objects in the scene translate or rotate parallel to the image plane, or when they approach or retreat parallel to the view direction [Brady and Horn, 1983; Terzopoulos, 1982]). The null-spaces \mathcal{N} of the semi-norms are simply the $M = \binom{d+m-1}{d}$ dimensional spaces of all polynomials over \mathbb{R}^d of degree less than or equal to $m-1$ [Schwartz, 1966, p. 60].

Under certain conditions $\mathcal{E}(v)$ becomes a norm in \mathcal{H} , which guarantees existence, uniqueness, and stability of the solution u to the variational principle. A possible set of conditions is that the I_i include evaluation functionals at an \mathcal{H} -unisolvent set of points (i.e., a set of M points which define a unique polynomial in the null space of the smoothness functional).

Several important visual reconstruction problems can be cast as spline approximation problems according to the foregoing variational formulation [Terzopoulos, 1982, 1984]. An equivalent point of view is that the ill-posed reconstruction problems are regularized by introducing a stabilizing functional which restricts admissible solutions to the class of generalized splines.

2.2. Multidimensional Tikhonov Stabilizers

In fact, it is readily possible to make an explicit connection between Tikhonov's stabilizers and the generalized spline functionals. Tikhonov's stabilizer of p -th order is the weighted Sobolev norm

$$\|v\|_p^2 = \sum_{m=0}^p \int_{\mathbb{R}^d} \lambda_m(\mathbf{x}) \left(\frac{d^m v(\mathbf{x})}{d\mathbf{x}^m} \right)^2 d\mathbf{x},$$

where the $\lambda_m(\mathbf{x})$ are given nonnegative continuous weighting functions [Tikhonov and Arsenin, 1977, pp. 69-70; Poggio and Torre, eq. 5]. The natural multidimensional generalization,

$$\|v\|_p^2 = \sum_{m=0}^p \int_{\mathbb{R}^d} \lambda_m(\mathbf{x}) \sum_{j_1, \dots, j_d=m} \frac{m!}{j_1! \dots j_d!} \left(\frac{\partial^m v(\mathbf{x})}{\partial x_1^{j_1} \dots \partial x_d^{j_d}} \right)^2 d\mathbf{x},$$

can be identified as a weighted summation of generalized spline functionals. If the smoothness functional $S(v)$ in the variational principle is taken to be the Tikhonov stabilizer, the admissible space \mathcal{H} becomes the p -th order Sobolev space.

3. Surface Approximation with Thin Plate Splines

In this section, we restrict our discussion of generalized approximation/regularization problems to two dimensions. Note that in two dimensions ($d=2$) the generalized spline functionals can be written as

$$|v|_m^2 = \int_{\mathbb{R}^2} \sum_{i=0}^m \binom{m}{i} \left(\frac{\partial^m v}{\partial x^i \partial y^{m-i}} \right)^2 dx dy.$$

Here they pertain directly to the problem of fitting surfaces to scattered data. This mathematical problem is of considerable concern in many application areas [Schumaker, 1976], and notably in vision to visible-surface reconstruction. Of the methods reviewed by Schumaker, the spline approximation techniques are of direct interest in our discussion.

An appealing spline approximation technique involves fitting to given data a surface function $u(x, y)$ which minimizes the functional

$$S_2(v) = \int \int v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2 dx dy$$

(see [Schumaker, 1976, section 3.4]). The functional will be recognized as an instance of the two-dimensional generalized splines for the case $m = 2$. It turns out that S_2 measures the (small deflection) bending energy of a thin plate (with zero Poisson ratio) [Courant and Hilbert, 1953]. Hence, Duchon [1976] refers to the surfaces which minimize $S_2(v)$ as thin plate splines.

Thin plate splines have seen several applications, including computer design of aircraft wings [Harder and Desmarais, 1972], automatic generation of digital terrain maps [Briggs, 1974], and variational analysis of meteorological fields [Wahba and Wendelberger, 1980].

In application to vision, Grimson [1981, 1983] employed $S_2(v)$ for the interpolation of visual surfaces from sparse disparity data (he referred to it as the "quadratic variation"). The relationship to thin plates was rediscovered by [Brady and Horn, 1983], and this physical model was developed further in [Terzopoulos, 1982, 1983a], where efficient, multiresolution surface reconstruction algorithms are devised.

3.1. Discontinuities & Thin Plate Surfaces Under Tension

The above applications demonstrate the basic usefulness thin plate spline surface approximation in the absence of discontinuities. Deficiencies of thin plate splines near discontinuities are particularly evident in the vision applications, however. The papers by Grimson and Terzopoulos contain a number of examples in which the reconstructed surface smoothes inappropriately across occluding boundaries and creases.

In the vicinity of discontinuities, a thin plate surface attempts to follow the sudden changes in the data, while it is simultaneously attempts to maintain its characteristic smoothness. It is consequently forced to overshoot the data points. This causes spurious inflection points (hence, ripples) near discontinuities. This phenomenon also manifests itself in classical spline interpolation [Ahlberg *et al.*, 1967].

Schweikert [1966] introduced splines under tension, which imitate the behavior of cubic interpolating splines, while suppressing the extraneous inflection points, which sometimes afflict cubic splines (see, also, [Cline, 1974]). Intuitively, tension can eliminate extraneous loops and ripples by reducing the length of the spline. Nielson [1974] characterizes splines under tension as interpolatory functions that minimize the functional $\int_a^b (\partial^2 v / \partial x^2)^2 dx + \alpha^2 \int_a^b (\partial v / \partial x)^2 dx$, where α is a constant to be chosen, called the tension. The first term influences smoothness, while the second term influences length. The basis functions for splines under tension involve exponentials. Nielson developed ν -splines, piecewise polynomial alternatives, one of whose advantages is that the tension can be set selectively at each interpolation point (see, also, [Barsky and Beatty, 1983]).

Pilcher [1974] suggested that the idea of splines under tension can be carried over to surfaces. His proposal involved restricting the area of a tensor product of polynomial splines, and he characterized this as a minimization problem whose solutions yield surfaces which he described as "elastic skins."

In [Terzopoulos, 1984] we proposed a new model for visible surfaces — the natural (physical) generalization of splines under tension to surfaces. Unlike Pilcher's elastic skins, our model involves no tensor products. The idea is to restrict the area of a thin plate spline by introducing surface tension.

It is well-known that membrane surfaces, in equilibria, minimize the functional

$$S_1(v) = \iint v_x^2 + v_y^2 dx dy,$$

which is the small deflection approximation of the surface area of v [Courant and Hilbert, 1953] (minimization of the true surface area leads to the famous Plateau's problem). $S_1(v)$ will be recognized as another instance of the two-dimensional generalized spline, this time for $m = 1$. Naturally, the membrane spline exhibits a lower order of smoothness than the thin plate spline.

Surface area can be restricted by combining the membrane spline functional $S_1(v)$ with the thin plate spline functional $S_2(v)$. We propose the convex combination

$$S_\tau(v) = \iint (1 - \tau) S_2(v) + \tau S_1(v) dx dy,$$

where the real valued parameter $\tau \in [0, 1)$ controls the tension. In view of the physical interpretation, solutions to the variational principle involving this functional will be referred to *thin plate surfaces under tension*.

As τ approaches 0, the thin plate surface under tension tends towards a normal thin plate spline, whereas as τ approaches 1, the surface area is increasingly restricted and the surface tends toward a membrane spline. Intermediate values of τ characterize a hybrid surface that amalgamates properties of the thin plate and the membrane. An appropriate value for the tension parameter can lead to advantages for surface reconstruction analogous to those offered by splines under tension over normal splines. For example, overshoots can be controlled, but at the cost of concentrating the curvature of the reconstructed surface near the constraints.

4. Controlled-Smoothness Stabilizers

Going one step further, the tension parameter in $S_\tau(v)$ can be made a nonnegative parametric function of two variables $\tau = \tau(x, y)$. This allows the tension to be manipulated over the domain of integration. Away from depth discontinuities, we can assign $\tau(x, y) = 0$ so that the surface acts as a thin plate. Along orientation discontinuities, we can assign $\tau(x, y) = 1$ so that the surface acts as a membrane, thus creasing freely. The smoothness functional $S_\tau(v)$ is deactivated entirely at occluding boundaries, where discontinuities in depth occur.

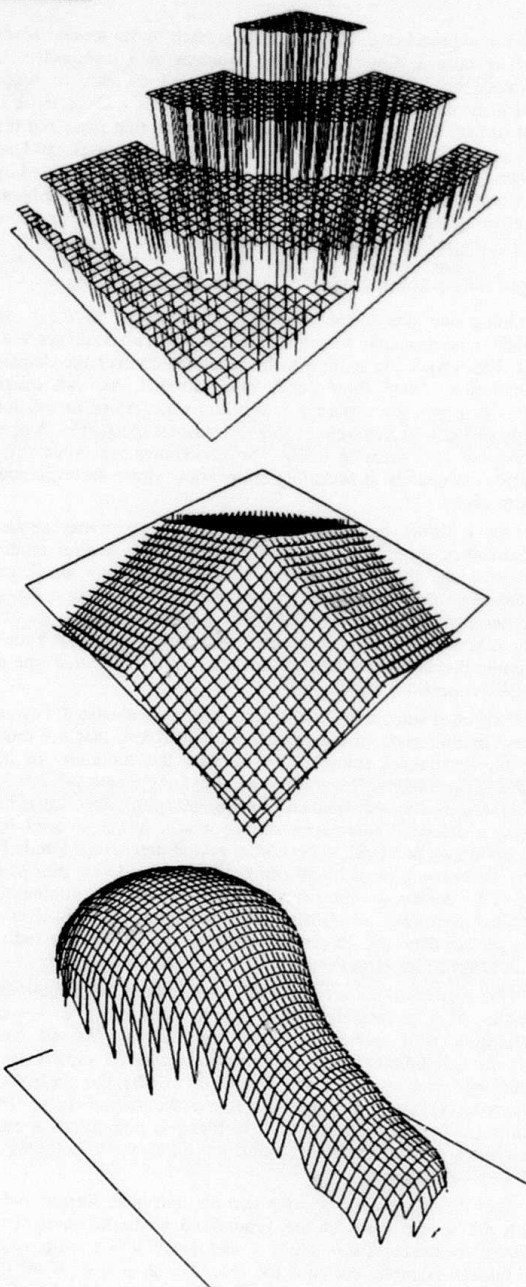
Fig. 1 shows examples of discontinuity preserving surface reconstructions using the thin plate surface under tension model. Interpreting the thin plate surface under tension as a stabilizing functional, we see that it provides local control over its smoothness properties. Specifically, it enables the reconstructed surfaces to "break" locally at known depth discontinuities and "crease" locally at known orientation discontinuities. We will refer to stabilizers of this type as *controlled-smoothness stabilizers*.

Controlled-smoothness stabilizers can be generalized beyond the case of thin plate surfaces under tension. Recall that the order m of the generalized splines $S_m(v)$ dictates the continuity of the admissible functions v . We mentioned that for the case $m = 1$ the solution can be characterized as a membrane spline. Physically, the membrane defines a continuous surface which, however, need not have continuous first (and higher) order partial derivatives. Similarly, for the case $m = 2$ the solution can be characterized as a thin plate spline. This defines a continuous physical surface with continuous first partial derivatives, which need not have continuous derivatives of degree greater than one. In general, physical solutions corresponding to S_m are functions possessing C^{m-1} continuity.

This suggests a convenient way of controlling the smoothness properties of a generalized spline stabilizer of order m — by combining it with splines of orders less than m . As we have shown, the multidimensional Tikhonov stabilizers do involve such compositions and, indeed, there is no reason why the parametric functions $\lambda_m(x)$ cannot play the same role as the tension $\tau(x, y)$. The functions can then be used to control the blending proportions of each component generalized spline S_m , thus regulating the smoothness of the Tikhonov stabilizer.

Both the spline under tension and the thin plate surface under tension are special cases of the generalized controlled-smoothness stabilizers, for the cases $d = 1, p = 2$, and $d = 2, p = 2$ respectively. As a further example, consider the case $d = 2, p = 3$. If all the $\lambda_m(x, y)$ are nonnegative functions, the solution specifies a surface spline which is constrained to have continuous second order partial derivatives (i.e. continuous curvature). However, it is possible to nullify this smoothness constraint by setting $\lambda_3(x, y) = 0$ at known curvature discontinuities in the x, y plane. This effectively reduces the functional to a thin plate spline locally, which need not have continuous curvature. Analogously, $\lambda_3(x, y)$ and $\lambda_2(x, y)$ can both be set to zero at known orientation discontinuities. Finally, all three λ functions can be set to zero at known depth discontinuities.

Figure 1. Discontinuity preserving surface reconstructions with the thin plate surface under tension. (Top) Reconstruction from scattered synthetic depth data with depth discontinuities. (Center) Reconstruction from synthetic orientation data showing depth and orientation discontinuities. (Bottom) Reconstruction of a light bulb from structured light data showing depth discontinuities.



4.1. Regularization Analysis & Discontinuity Detection

Controlled-smoothness stabilizers are applicable to ill-posed visual reconstruction problems in which the locations and orders of discontinuities in the solutions are known or can be determined in advance. An interesting question arises: Is it possible to determine the location of discontinuities as an integral part of regularization analysis? We offer some preliminary ideas on how to do this in view of the development in this paper.

The detection of discontinuities can be incorporated formally into a variational principle whose stabilizing functional $S(v)$ is a controlled-smoothness stabilizer. The basic problem of finding the optimal solution u is then augmented with the additional requirement to find $p+1$ optimal parametric functions $\lambda_m(x)$, $m = 0, 1, \dots, p$, which determine the locations of discontinuities up to order p .

This larger problem is made more difficult by the fact that it is non-convex. One way to approach it is by solving a chain of easier convex problems, where the (improving) solution at each stage becomes data for the subsequent problem. This is essentially the discontinuity detection approach implemented in [Terzopoulos, 1984]. Blake [1983] describes an alternative graduated non-convexity method for finding discontinuities in images. He has implemented it for the highly restricted case of jump discontinuities between piecewise constant regions. While these approaches are not guaranteed to yield the true global optimum, they typically produce good local optima. A possibility for finding the global optimum is the annealing-based technique for finding discontinuities described by [Geman and Geman, 1984]. This approach, while computationally expensive on sequential machines, nonetheless appears to be effective [Marroquin, 1984].

Optimality criteria for the parametric functions should be made part of the energy functional $E(v)$. A necessary criterion restricts the number of discontinuities placed, and other potentially useful criteria for placement of discontinuities may include a preference for piecewise smooth discontinuity contours. For the two-dimensional case of surfaces, a natural formulation for the optimal placement of discontinuities may naturally reduce to a one-dimensional ill-posed reconstruction problem. In such a case, analogous controlled-smoothness regularization can be applied in a single dimension, yielding a one-dimensional spline problem.

5. Conclusion

Classical regularization methods can be deficient in application to ill-posed visual reconstruction problems involving discontinuities. To remedy the situation, the smoothness properties of standard spline stabilizing functionals must be relaxed locally at discontinuities.

Exploring the close connection of regularization analysis and generalized spline approximation, we were led to interpret the natural multidimensional generalizations of Tikhonov's stabilizers as controlled-smoothness splines. The smoothness properties of these combinations of generalized spline functionals can be regulated locally through a set of parametric functions. The controlled-smoothness spline functionals encompass the special cases of classical splines under tension, as well as thin plate surfaces under tension. The latter were developed as physical models in our recent work on visible-surface reconstruction, a fundamentally ill-posed visual problem. Controlled-smoothness regularizers suggest possibilities for incorporating discontinuity detection into regularization analysis. This is a challenging problem that is currently under investigation.

Acknowledgements

I thank Tomaso Poggio for discussions which led to the writing of this paper and for his discerning comments on a draft. Financial support provided by the Natural Sciences and Engineering Research Council of Canada, Ottawa, and the Fonds F.C.A.C., Québec, Canada, is gratefully acknowledged. This paper describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's Artificial Intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643, the Office of Naval Research under contract number N0014-80-C-0505, and the System Development Foundation.

References

- Anieng, J.H., Nilson, E.N., and Walsh, J.L., [1967], *The Theory of Splines and their Applications*, Academic Press, New York.
- Barrow, H.G., and Tenenbaum, J.M., [1978], "Recovering intrinsic scene characteristics from images," *Computer Vision Systems*, A.R. Hanson and E.M. Riseman (ed.), Academic Press, New York, 3-26.

- Barsky, B.A., and Beatty, J.C., [1983]. "Local control of bias and tension in beta-spline." *ACM Trans. Graphics*, 2, 109-134.
- Blake, A., [1983]. Parallel computation in low-level vision, Ph.D. thesis, Department of Computer Science, University of Edinburgh, Edinburgh, Scotland.
- Brady, J.M., and Horn, B.K.P., [1983]. "Rotationally symmetric operators for surface interpolation," *Computer Vision, Graphics, and Image Processing*, 22, 70-94.
- Briggs, I.C., [1974]. "Machine contouring using minimum curvature," *Geophysics*, 39, 39-48.
- Cline, A.K., [1974]. "Scalar- and planar-valued curve fitting using splines under tension," *Comm. ACM*, 17, 218-220.
- Courant, R., and Hilbert, D., [1953]. *Methods of Mathematical Physics*, Vol. I, Interscience, London.
- Duchon, J., [1976]. "Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces," *R.A.I.R.O. Analyse Numérique*, 10, 5-12.
- Duchon, J., [1977]. "Splines minimizing rotation-invariant semi-norms in Sobolev spaces," *Constructive Theory of Functions of Several Variables*, A. Dodd and B. Eckmann (ed.), Springer-Verlag, Berlin, 85-100.
- Geman, S., and Geman, D., [1984]. "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Trans. Pattern Analysis & Machine Intelligence*, to appear.
- Grimson, W.E.L., [1981a]. *From Images to Surfaces: A Computational Study of the Human Early Visual System*, MIT Press, Cambridge, MA.
- Grimson, W.E.L., [1983]. "An implementation of a computational theory of visual surface interpolation," *Computer Vision, Graphics, and Image Processing*, 22, 39-69.
- Harder, R.L., and Desmarais, R.N., [1972]. "Interpolation using surface splines," *Jour. Aircraft*, 9, 189-191.
- Laurent, P.J., [1972]. *Approximation et Optimisation*, Hermann, Paris.
- Marr, D., [1982]. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, Freeman, San Francisco, CA.
- Marroquin, J.L., [1984]. Boundary preserving smoothing and interpolation, MIT A.I. Lab., Cambridge, MA. AI Memo No. 792.
- Meinguet, J., [1979]. "Multivariate interpolation at arbitrary points made simple," *Jour. Applied Math. and Physics (ZAMP)*, 30, 292-304.
- Nielson, G.M., [1974]. "Some piecewise polynomial alternatives to splines under tension," *Computer Aided Geometric Design*, R.E. Barnhill and R.F. Riesenfeld (ed.), Academic Press, New York, 209-235.
- Pileher, D., [1974]. "Some piecewise polynomial alternatives to splines under tension," *Computer Aided Geometric Design*, R.E. Barnhill and R.F. Riesenfeld (ed.), Academic Press, New York, 237-253.
- Poggio, T., and the staff, [1984]. "MIT progress in image understanding," *This proceedings*.
- Poggio, T., and Torre, V., [1984]. Ill-posed problems and regularization analysis in early vision, MIT A.I. Lab., Cambridge, MA. AI Memo No. 773, reprinted in this proceedings.
- Reinsch, G., [1967]. "Smoothing by spline functions," *Numer. Math.*, 10, 177-183.
- Schoenberg, I.J., [1964]. "Spline functions and the problem of graduation," *Proc. National Academie of Sciences*, 52, 947-950.
- Schwartz, L., [1966]. *Théorie des Distributions*, Hermann, Paris.
- Schweikert, D.G., [1966]. "An interpolation curve using a spline in tension," *J. Math. and Physics*, 45, 312-317.
- Schumaker, L.L., [1976]. "Fitting surfaces to scattered data," *Approximation II*, G.G. Lorentz, C.K. Chui, L.L. Schumaker (ed.), Academic Press, New York, 203-267.
- Terzopoulos, D., [1982]. Multilevel reconstruction of visual surfaces: Variational principles and finite element representations, MIT A.I. Lab., Cambridge, MA, 1982, AI Memo No. 671, reprinted in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (ed.), Springer-Verlag, New York, 1984, 237-310.
- Terzopoulos, D., [1983a]. "Multilevel computational processes for visual surface reconstruction," *Computer Vision, Graphics, and Image Processing*, 24, 52-96.
- Terzopoulos, D., [1983b]. "The role of constraints and discontinuities in visible-surface reconstruction," *Proc. 8th Int. J. Conf. AI, Karlsruhe, W. Germany*, 1073-1077.
- Terzopoulos, D., [1984]. Multiresolution computation of visible-surface representations, Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA.
- Tikhonov, A.N., [1963]. "Solution of incorrectly formulated problems and the regularization method," *Soviet Math. Dokl.*, 4, 1035-1038.
- Tikhonov, A.N., and Arsenin, V.A., [1977]. *Solutions of Ill-Posed Problems*, Winston & Sons, Washington, DC.
- Wahba, G., and Wendelberger, J., [1980]. "Some new mathematical methods for variational objective analysis using splines and cross validation," *Monthly Weather Review*, 108, 1122-1143.

SPATIAL REASONING FROM LINE DRAWINGS OF POLYHEDRA

Thomas M. Strat

SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025

Abstract

A method is presented for transforming a set of line drawings of a polyhedral scene into a representation that embodies the three-dimensional structure of the scene. The line drawings are first converted to machine-readable form and then back-projected to acquire a wire frame skeleton of the scene. A novel three-dimensional constraint propagation scheme is then employed to transform the wire frame to a description of the solid objects which compose the scene. This process has applications in computer-aided design as well as in machine understanding of multiple images. The paper concludes with a discussion of issues related to achieving the same result from a single view.

1. Introduction

Machines that must reason about or function in a three-dimensional world must be equipped with models of objects in that world. A multitude of representations has been devised for three-dimensional models [1], yet the specification of individual models can be a tedious undertaking. This paper examines methods for computing a three-dimensional model of a particular class of objects from a particular form of input—polyhedral objects from line drawings.

Researchers in computer-aided design have produced numerous systems that manipulate models of solid objects to assist in the design, analysis, or fabrication of everything from machine parts to factories. The act of specifying a model is one of the most difficult tasks associated with these systems.

In an interactive image-understanding system, there are several sources of line drawings. One can envision a very competent line-finder that automatically extracts the line drawings of selected objects. Alternatively, the user can specify the lines in an image by pointing at their endpoints with a mouse or other input device. A third possibility is for the user to draw the figures freehand or with mechanical assistance. Whatever the means of entry, the objective is to produce a three-dimensional sketch that captures the volumetric nature of the objects.

This paper is concerned with deriving the representation geometrically, as opposed to using model-based representations. It is divided into two parts: The first presents an algorithm for deriving a volumetric description of a polyhedral scene when mul-

multiple views are available; the second part explores the problem of accomplishing this task when only one line drawing is available.

2. Multiple Views

The algorithm to be described solves for a three-dimensional description of a scene when several views are available. The overall process can be thought of as accepting a set of line drawings of a scene as input and providing as output a display of the object from any angle, with all hidden lines removed. The algorithm consists of four sequential modules called input, projection, wire frame, and display.

The required input is a set of two or more line drawings and the angular relationships among them. A line drawing is restricted to be the projection (orthographic or perspective) of a polyhedron from a particular vantage point and, as a result, is a collection of straight line segments.

The input module is responsible for producing a data structure that specifies the positions of all lines and their endpoints in a line drawing. Its actual form will vary with the source of the line drawing, as different input processes dictate different procedures for constructing the data structure.

The projection module computes the three-dimensional coordinates of vertices and edges that may have given rise to the endpoints and lines in the drawings. The output of the projection module is in the form of a three-dimensional wire frame, which is represented as a list of vertices and edges. The computation is carried out by back-projecting the points in each line drawing and determining their points of intersection.

Next in the pipeline is the wire frame module, which is the most interesting of the four. Its task is to derive the solid object that corresponds to the given wire frame. It employs a Waltz-style constraint propagation scheme [8], but differs significantly by assigning labels to spatial regions and propagating them throughout the three-dimensional structure, in contrast with propagation across a two-dimensional line drawing. Only two labels are allowed (SOLID and HOLE), and a consistent labeling is usually achieved very quickly.

The display module uses the labeled output of the wire frame module to produce a display of the object, with hidden lines removed. As will be seen shortly, the hidden-line algorithm is somewhat unusual in the way it takes advantage of the label information in the wire frame.

The research reported herein was supported by the Defense Advanced Research Projects Agency under Contract No. MDA 903-83-C-0027.

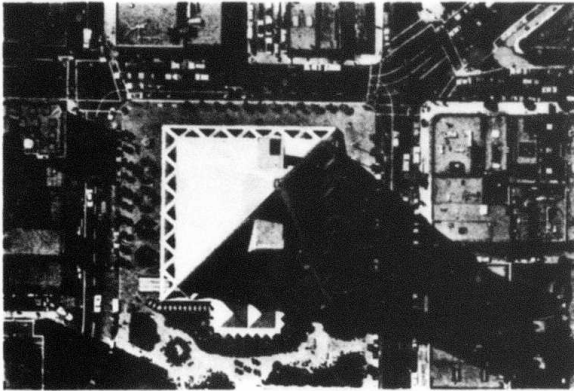


Figure 1: Photographs of the Transamerica Building

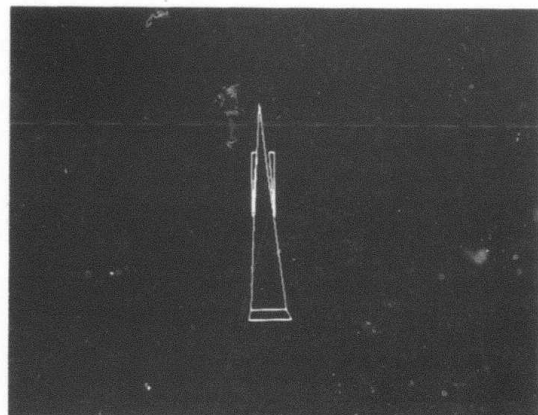
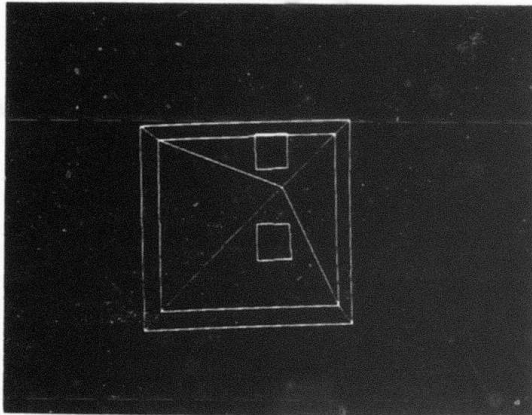


Figure 2: Line Drawings of the Transamerica Building

2.1. The Input Module

The input module is actually a set of alternative modules. The one to be used depends on the source of the line drawing. In all cases, a **line drawing** is defined to be composed of a set of line segments, to be referred to as **lines**, and their points of intersection, to be referred to as **endpoints**. Lines are restricted to intersect only at their endpoints. No curves are permitted since the line drawing is assumed to be the projection of a polyhedron. Furthermore, all edges of an object, visible or not, must appear in all line drawings. That is, hidden lines, which are normally represented as dashed lines in engineering drawings, are to be depicted like any other line, since the algorithm does not accord them any special treatment.

The most direct means for specifying the line drawing would be to provide the coordinates of the endpoints explicitly. Another procedure is envisioned that allows specification of line drawings in a more natural way.

In one scenario, the user will sketch or trace a line drawing directly into the system by means of a pointing device, such as a mouse or graphics tablet. Inaccuracies inherent in this procedure must be resolved before the data are passed to the projection module. Sugihara presents a method for identifying incorrect line drawings and correcting vertex position errors [7]. Regardless of the method used, the input module provides a data base con-

taining the endpoints, lines, and viewpoint for each line drawing. As an example, Figure 1 shows two images of the Transamerica Building in San Francisco. The line drawings of Figure 2 were obtained by tracing the edges of the building with a mouse-controlled cursor. The camera models of each image were computed on the basis of ground truth data obtained from a map of San Francisco.

2.2. The Projection Module

Given the data base specifying a set of line drawings of a scene and the associated camera models, the projection module determines the wire frame of the scene that could have given rise to those drawings. A **wire frame** is a set of vertices and edges, where an **edge** is the intersection of two faces of an object, and a **vertex** is the intersection of two edges. The algorithm follows closely that of Wesley and Markowsky [9].

In the first phase, the vertices of the wire frame are computed. Figure 3 illustrates the geometry involved. Any vertex of the wire frame is constrained to lie on the line connecting the viewpoint and the endpoint that is the projection of the vertex in the line drawing. Such lines are constructed for every endpoint in every line drawing. The intersections of these lines are computed and entered as vertices of the wire frame. Note that this procedure may find vertices that are not in fact vertices of the

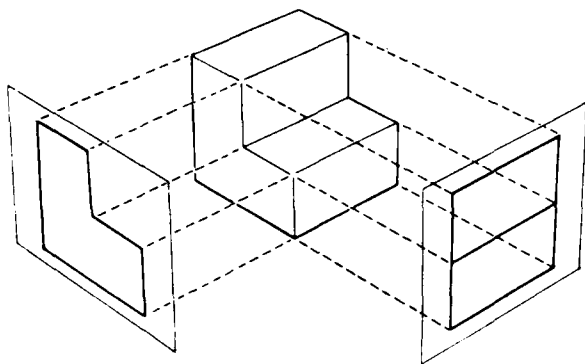


Figure 3: The geometry of backward projection.

wire frame. Some of these vertices are identified in later stages of processing and discarded; the remainder are the result of alternative legal interpretations of the line drawings. It is also possible that some real vertices may be missed, but, fortunately, they can be found during the second phase of the projection module's operation.

In Phase two, the edges of the wire frame are found. Two vertices are connected by an edge only if that edge is consistent with all views provided. An edge is consistent with a view if that edge projects to a line in the view, to a set of continuous colinear lines, or to a single endpoint. When all such edges have been found, the edges are checked for internal intersections. Any such intersections are the missing vertices of Phase one and are added to the data base. Just as extra vertices may have been found earlier, extra edges may arise for the same reasons.

In Phase three, any vertex with fewer than three incident edges is eliminated. (Realizable solid objects always have at least three edges meeting at any vertex.) Any accompanying edges are also removed and the pruning is continued until a stable configuration is reached. Usually, however, there are no vertices to be removed in this manner.

At this point, a wire frame has been computed that is guaranteed to encompass all the edges and vertices of the object. If the set of line drawings provided determines the object uniquely, the wire frame will correspond exactly to the wire frame of the object. If the line drawings are ambiguous, the wire frame may contain vertices and edges present in one interpretation but absent in others. The ambiguous case can be accommodated by invoking the wire frame module for each of the possible interpretations. Those found to be inconsistent can be disregarded; those found to have a legal interpretation can be construed as alternative solutions. The remainder of this paper assumes that the wire frame has been determined uniquely.

2.3. The Wire Frame Module

The input to the wire frame module is a data structure representing a wire frame that contains only edges and vertices that correspond to true edges and vertices of the underlying scene. The module's job is to find out which regions are occupied by

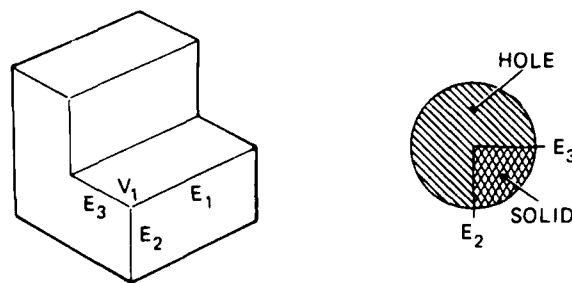


Figure 4: Spoke diagram of edge E_1 .

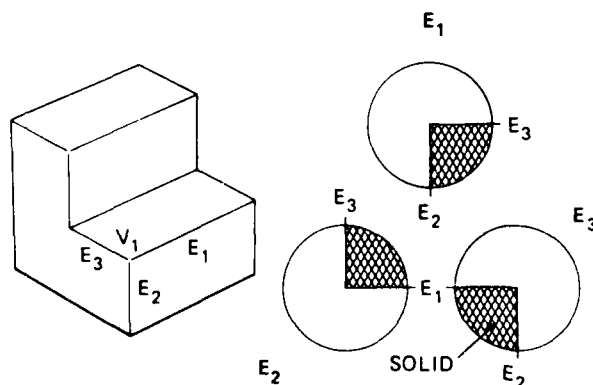


Figure 5: Propagation of Intravertex Constraints.

solid matter and which are not, relative to the wire frame. Its basic tool for performing this reasoning is the spoke diagram (Figure 4). The spoke diagram is an edge-on view of a vertex. The spokes are the projections of the edges at a vertex onto the plane that is perpendicular to the selected edge. The spoke diagram in the figure is the view along edge E_1 toward vertex V_1 , such that E_1 itself projects out of the drawing. The sector between two spokes represents the solid angle defined by the two edges corresponding to the two spokes and the selected edge. The solid angle must either be filled completely with matter or be completely void of matter, because boundaries between matter and space can occur only at faces and all faces are bounded by edges. Therefore, each sector can be labeled by either SOLID or HOLE to reflect this choice. The task of the wire frame module is then to assign a label of SOLID or HOLE to every such solid angle, as defined by the wire frame.

As mentioned earlier, the wire frame module is a constraint propagation algorithm. Three separate processes serve to constrain and propagate the labelings:

1. **Intravertex constraints**—These serve to propagate labels (SOLID or HOLE) among the spoke diagrams at a given vertex, as in the example of Figure 5. Here an assignment of SOLID to the sector between the spokes corresponding to edges E_2 and E_3 , in the spoke diagram of E_1 at

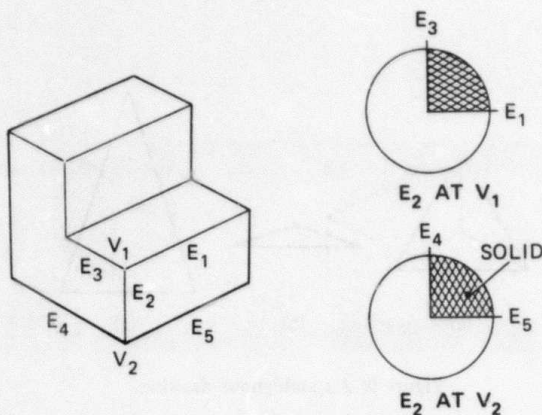


Figure 6: Propagation of Intervortex Constraints.

vertex V_1 , can be propagated to constrain the other spoke diagrams at V_1 . In the spoke diagram of edge E_2 , the sector between edges E_1 and E_3 must also be labeled **SOLID**. The same applies to edge E_3 . Corresponding sectors must be labeled identically because they are representations of the same volume in space. Care must be taken to account for sectors wider than 180° .

2. **Intervortex constraints**—These serve to propagate labels from one vertex to another, as in Figure 6. Here the newly assigned label of **SOLID** to the sector between E_1 and E_3 , in the spoke diagram of edge E_2 at vertex V_1 , can be propagated along E_2 to constrain the spoke diagrams at V_2 . In this case the sector between E_4 and E_5 must also be **SOLID**. The two pairs of edges define a dihedral angle along E_2 . This type of constraint propagation is always valid because the state can change between **SOLID** and **HOLE** only at a face. If a face occurred somewhere along E_2 , it would intersect with E_2 at a point other than one of its endpoints, which is precluded by the definition of a wire frame.
3. **Vertex Labeling Constraints**—Some potential labelings of a spoke diagram are not legal. It is conceivable to determine the set of legal labelings for trihedral vertices, tetrahedral vertices, etc., but in practice this becomes unwieldy. As implemented, the wire frame module applies a somewhat weaker constraint. It prohibits any labeling that is all **SOLIDS** or all **HOLEs**. Since such an assignment would render the edge nonexistent, it could not be a legal interpretation of the wire frame. In practice, this rather weak constraint has generally proved satisfactory.

Unfortunately, there are numerous special cases that arise during execution. The intravertex constraints must distinguish between angles greater or less than 180° to ensure proper labeling. The intervortex constraints must be applied properly when the spoke diagrams at each endpoint do not coincide. Extra spokes and missing spokes are two such cases.

The wire frame module is a control structure for propagating these three constraints throughout a wire frame. For efficiency, the implementation actually applies the three constraints simultaneously. It includes checks for completion and inconsistencies. Although a given wire frame may be ambiguous, (i.e., allow more than one interpretation), the algorithm is guaranteed to termi-

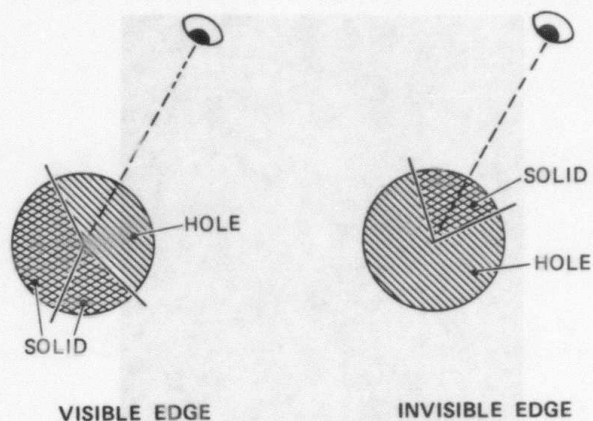


Figure 7: Hidden-line removal.

nate. Each step assigns a label, does nothing, or detects an inconsistency and quits. Once a label is assigned, it is never removed. The algorithm proceeds until all sectors have labels or no change has been detected through a complete iteration. If some sectors remain unlabeled, the algorithm is continued separately for each possible labeling (**SOLID** or **HOLE**) until a completely labeled wire frame is obtained.

2.4. The Display Module

The display module uses the labels computed by the wire frame module to eliminate lines hidden from view. Most of the lines to be eliminated can be readily identified by the process illustrated in Figure 7. For each edge, the direction to the viewpoint is computed and that ray is superimposed on the spoke diagram. If the ray pierces a **SOLID** sector, that edge is a rearward-facing edge and is not displayed. If the ray falls into a **HOLE** sector, further processing is needed. The projection of the edge is checked for intersection with the projection of all other visible edges. If no such intersection exists, the edge is displayed. If an intersection is found, the intersecting edge is examined to determine which side of the edge is occluded (or if both are). The edge is split at its point of intersection and each part is handled in the same manner recursively. It should be noted that this algorithm will fail to eliminate certain occluded edges if their projections do not intersect with any other projected edges. A less efficient search would be required to eliminate these.

Returning to the example of the Transamerica Building of Figure 1, a wire frame was obtained from the line drawings and was subsequently processed by the wire frame module. Figure 8 shows a perspective view of the result, with hidden lines removed, indicating realization of the correct wire frame and the successful assignment of solid material relative to it.

2.5. Summary

The class of objects that may be modeled is not as restrictive as it may seem at first glance. Any polyhedron is permissible and may contain arbitrary concavities and holes. Curves may be approximated by a number of line segments. Several polyhedra may be juxtaposed in any manner.

The line drawings may be either orthographic or perspective, and from any vantage point. Accidental alignments pose no par-

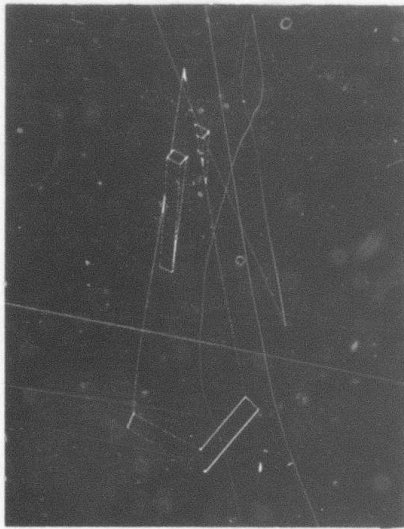


Figure 8: Hidden-line removal on the Transamerica Building.

ticular problem. A unique wire frame will be constructed if any drawing is in general position (i.e., no vertices or edges coincide in the projection). Reconstruction of a unique wire frame is also guaranteed if corresponding lines in each view are designated.

Just as a set of line drawings may define more than one wire frame, some wire frames may define more than one object. The wire frame module will derive all possible interpretations when confronted with an ambiguous wire frame.

Stronger vertex-labeling constraints may be necessary to assure the correct interpretation of some objects. Further testing is required to refine the constraints and demonstrate the ability to derive volumetric descriptions of a wide range of polyhedral scenes.

3. Single View

We have described a procedure that makes it possible to infer the shape of an object from several views. A means for accomplishing this task from a single view is also desirable. After all, humans appear to have little trouble constructing a three-dimensional representation of an arbitrary object from a single line drawing. While this section does not offer a method for approximating human performance, it does point to some promising approaches toward that objective.

The most relevant early work on polyhedral-scene interpretation is by Mackworth [6]. His program, POLY, has provided a framework for subsequent shape-from-line-drawing methods. POLY first achieves a qualitative interpretation of a line drawing by parsing the line segments to ascertain which of them are convex, concave, or occluding. The convex and concave edges yield constraints that can often be used to determine the orientation of the polyhedral faces quantitatively. By using the now familiar technique, the orientations of faces that meet at an edge are restricted to lie on a line in gradient space that is perpendicular to the image line of that edge. Combining these constraints through triangulation often yields the orientations of all the faces.

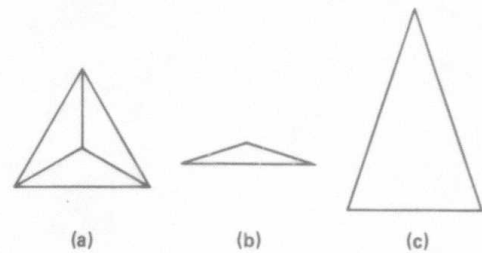


Figure 9: An ambiguous drawing.

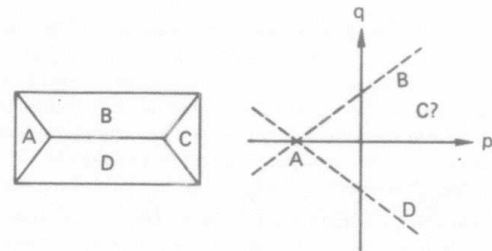


Figure 10: Another ambiguous drawing.

Unfortunately, POLY has many shortcomings that limit its competence, as pointed out by Draper [3]. He presents methods for overcoming some of POLY's limitations, yet human-level performance is still beyond reach.

One observation that dampens hope of ever producing an algorithmic solution to the problem is that human perception of line drawings can be extremely subjective. Figure 9(a) shows a triangular pyramid that can be variously interpreted as either very flat (b), or very pointed and elongated (c). In gradient space, this phenomenon manifests itself in the fact that the scale of the gradient space cannot be resolved from a line drawing algebraically without higher-level assumptions. In fact, neither the scale nor the origin of gradient space can be determined by existing methods that utilize the gradient-space representation.

Recent work by Barnard [2] addresses this issue of subjective interpretation. For instance, if one is willing to assume that the pyramid in Figure 9 is actually composed of mutually perpendicular faces, that information can be used to specify the orientations of all faces of the object uniquely. Moreover, what this accomplishes is to establish both the origin and scale of gradient space. Barnard's procedure only requires identification of three lines in the scene known to be orthogonal (they need not intersect) in order to compute these quantities. This represents a powerful tool when coupled with the purely objective approach to the interpretation of line drawings. The requirement for finding a set of mutually orthogonal lines is no great burden when many scenes of man-made objects are being examined, but one would prefer a purely general method for the subjective interpretation of line drawings. Heuristics exploiting parallelism, symmetry,

and compactness of description may provide a useful inroad.

The determination of the origin and scale of gradient space is in itself not sufficient for the interpretation of all faces in a polyhedral scene. Figure 10 shows an object and its gradient-space interpretation. Even if the orientation of face *A* is known exactly, the locations in gradient space of faces *B*, *C*, and *D* are still underdetermined. The figure is analytically ambiguous but subjectively resolvable; additional heuristics may be necessary to find the solution.

Another issue inherent in the analysis of line drawings is how best to cope with imprecise input. Line drawings may be extracted from real images or may be hand-drawn. One would prefer an algorithm that does not degenerate completely when confronted with inaccurate drawings. A drawing of an "impossible" object, that is, a drawing that does not correspond to any geometrically possible object, should be interpreted as the "closest" object that is geometrically permissible. Kanade's algorithm [4], which works through iterative minimization of errors, provides a framework for achieving this goal. One can conceive of designing a system that theoretically supports only orthographic line drawings, and using it to interpret perspective drawings. If the focal length is sufficiently large, the perspective distortion might be treated as drawing error and an approximate interpretation obtained. While the validity of this approach depends on the application in mind, it does circumvent the difficulties of a truly perspective model.

The gradient-space representation is unsuitable for analyzing perspective drawings [5]. The primary reason is the inability to capture the concept of sidedness of a plane in gradient space. Sidedness reasoning is essential to the interpretation of perspective drawings because either side of a plane may be visible, depending on the plane's location in a perspective drawing. Formalisms based on the Gaussian sphere overcome this problem. The mathematics becomes a little more complex (quadratic versus linear equations), but the two solutions to each quadratic equation, corresponding to the two sides of a plane in three-dimensional space, enable quantitative analysis of perspective scenes.

4. Summary

Recovering the shape of an object from a single line drawing of that object is a difficult problem. Further investigation is necessary to achieve human-level competence.

The algorithm presented for interpreting scenes from multiple views embodies a novel approach to a long-standing problem. The type of spatial reasoning used promises to be applicable in other situations as well. The technique may be successful when only a portion of an object is visible and may perform adequately even with inaccurate line drawings (such as those missing a line here or there). It is a local reasoning process that may be especially appropriate for supporting higher-level reasoning about solid objects.

REFERENCES

- [1] Baer, A., Eastman, C., and Henrion, M., "Geometric Modeling: a Survey", *Computer Aided Design*, Vol. 11, September 1979, pp. 253-272.
- [2] Barnard, S. T., "Interpreting Perspective Images", Tech-

nical Note 271, Artificial Intelligence Center, SRI International, Menlo Park, California, November 1982.

- [3] Draper, S. W., "The Use of Gradient and Dual Space in Line-Drawing Interpretation" *Artificial Intelligence* 17, August 1981, pp. 461-508.
- [4] Kanade, T., "Recovery of the 3-D Shape of an Object from a Single View", *Artificial Intelligence* 17, August 1981, pp. 409-460.
- [5] Kender, J. R., "Shape from Texture", CMU-CS-81-102, Carnegie-Mellon University, November 1980.
- [6] Mackworth, A. K., "Interpreting Pictures of Polyhedral Scenes", *Artificial Intelligence*, 4, 1973, pp. 121-137.
- [7] Sugihara, K., "Mathematical Structures of Line Drawings of Polyhedrons— Toward Man-Machine Communication by Means of Line Drawings", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 5, September 1982, pp. 458-469.
- [8] Waltz, D., "Understanding Line Drawings of Scenes with Shadows", *The Psychology of Computer Vision*, P.H. Winston, Ed., McGraw-Hill Book Co., Inc., New York, 1975, pp. 19-91.
- [9] Wesley, M.A. and Markowsky, G., "Fleshing Out Projections", *IBM J. Res. Develop.*, Vol. 25, No. 6, November 1981, pp. 934-954.

COMPUTING DENSE DISPLACEMENT FIELDS WITH CONFIDENCE MEASURES IN SCENES CONTAINING OCCLUSION

P. Anandan

Computer and Information Science Department
University of Massachusetts
Amherst, MA 01003

ABSTRACT

Matching successive frames of a dynamic image sequence using area correlation has been studied for many years by researchers in machine vision. Most of these efforts have gone into improving the speed and the accuracy of correlation matching algorithms. Yet, the displacement fields produced by these algorithms are often incorrect in homogeneous areas of the image and in areas which are visible in one frame, but are occluded in the succeeding frames. Further, these displacement fields are often incorrect even at non-occluded areas that border occlusion boundaries. In this paper, we present a confidence measure which indicates the reliability of each displacement vector computed by a specific hierarchical correlation matching algorithm. We also provide an improved hierarchical matching algorithm which performs particularly well near occlusion boundaries. We demonstrate these with experiments performed on real image sequences taken in our robotics laboratory. A more detailed version of this work appears in [Anan84].

1. INTRODUCTION

One of the powerful techniques that have been studied by researchers in image processing and computer vision for the purpose of matching images is area correlation [Agga81a, Bard80, Hann74, Mora81, Genn80, Burt83, Glas83, Wong78, Lawt84]. Much of this work has addressed issues in choosing a useful match measure, increasing the accuracy of the match, and in reducing the computational complexity of the matching algorithms. However, most of the current techniques produce false matches when applied to scenes containing occlusion, i.e., where a certain area of the image which is visible in one frame is hidden by other moving areas in the succeeding frames. The problem include the following:

- The search required may be large.
- The spatial resolution of the displacement field produced by correlation matching becomes poorer as the size of the sample window increases [Genn80].

- Correlation matching tends to produce false matches in areas of the image where the variation is low.
- Correlation matching fails most miserably in areas of the image that are occluded and the non-occluded areas that border them.

Some researchers have used hierarchical search strategies [Glas83, Burt83, Wong78] to reduce the amount of search required. However, the problems in processing scenes containing occlusion still remain. In this study, we use the hierarchical matching algorithm of Glaser, Reynolds, and Anandan [Glas83] as our basis. We isolate the situations where this matching algorithm fails by computing a confidence measure which estimates the reliability of each displacement vector that is computed by the matching algorithm. We then modify the hierarchical search strategy to improve the results, especially near occlusion boundaries. The result is a computationally efficient matching algorithm which provides a dense displacement field with estimates of reliability of each displacement vector.

Section 2 of this paper describes the various types of correlation techniques that have been investigated by researchers. Section 3 describes some of the work done by other researchers for finding a confidence measure, and describes the measure chosen for this work. Section 4 describes our modifications to the search strategy. Section 5 describes some applications and the possible future directions of this research.

2. TYPES OF CORRELATION MEASURES AND ALGORITHMS

A variety of correlation measures and associated search strategies have been studied by researchers in the field of image-matching. In this section we briefly review some of these measures and some of these search strategies. The particular choices of the measure and the search strategy are not always independent of each other. In our discussion below, we point out such dependencies where they occur.

2.1 Types of Correlation

Some typical correlation match measures that are used

by researchers are described in [Hann74]. These include direct correlation, mean normalized correlation, variance normalized correlation, sum of the squares of the differences between corresponding pixels (SSD), and sum of the magnitudes of the differences.

A comparative study of direct, mean-normalized and variance-normalized correlation measures can be found in [Burt82]. In addition, Burt also suggests the use of Laplacian filtered images for matching. Although this filtering process can be used in combination with any of the above match measures, his study includes only the Laplacian-filtered direct correlation. Burt shows that the most reliable results are consistently obtained by choosing correlation with both mean and variance normalizations. However, this process (especially, variance normalization) is computationally expensive. Therefore, he recommends the computationally efficient Laplacian-filtered direct correlation which appears relatively insensitive to both mean and contrast changes between the images, although this measure performs poorly in the presence of high frequency noise.

The reason for the success of the Laplacian-filtered correlation process is that the mean value of a Laplacian-filtered image tends toward zero as the sample window size increases. Thus, the filtering process has the effect of mean-normalizing the correlation values. However, we found that when the window sizes are smaller than 8×8 , the mean of a sample window, though small, was not nearly zero. In such cases using Laplacian-filtered SSD provides more accurate results than Laplacian-filtered correlation. This is because the SSD measure is sensitive to the difference of the means of the two areas that are compared. For smaller windows,

the differences of the means tend to be small, even though the means are not zero. Our own experiments suggest (see Table 1) that for a fixed size of the matching window, the performance of Laplacian-filtered SSD exceeds that of the Laplacian-filtered correlation using the same sized window. Based on this, we have used Laplacian-filtered SSD as the match measure for the rest of our study.

Although the problem of mean changes can be eliminated by using Laplacian-filtered SSD, there are still two sources of false matches. First, contrast variations between the images degrades the performance of the SSD measure [Hann74]. Second, the Laplacian-filtering process removes image variations below a certain frequency, thus causing the image structures to repeat beyond a distance corresponding to the filter cut-off wavelength. Therefore, if the search area is large, there is greater potential for the occurrence of false matches. In the following section, we note that the search strategy used can be helpful in alleviating both these difficulties.

2.2 Search Strategies

There are only a limited number of search strategies that have been employed by various researchers. The most obvious strategy is to search the whole area within the expected maximum displacement. Hannah [Hann74] and Gennery [Genn80] use this technique, although Gennery mentions a number of ways to cut down the computational cost of the search, and mentions the possibility of using global techniques to get approximate estimates which are then improved using local searches. Lawton [Lawt84] uses a search technique that is most suitable for the case of pure translational motion of the camera. In this case a global search is performed for the focus of expansion (FOE), which is the intersection of the translational axis and the image plane. Specific values of the FOE are evaluated and for each, the local searches for optimal feature matches are constrained to lie along radial lines emanating from the assumed FOE.

Wong and Hall [Wong78], Glazer et. al. [Glas83], Burt et. al. [Burt83], and Moravec [Mora81] all use a multi-resolution coarse-fine strategy, but there are important differences among them. Among these differences, we are interested in the fact that Wong and Hall and Moravec used low-pass filtered images, whereas Glazer et. al., and Burt et. al. used band-pass filtered images. Burt et. al. used a strategy where the searches at the different levels of resolution operate independently of each other. This results in low-frequency coarse resolution searches detecting large displacements and higher-frequency finer resolution searches detecting smaller displacements in the image. Glazer et. al. [Glas83] use a strategy which utilizes the approximate estimate given by the low-frequency, coarse resolution searches as a starting value to define the search in the higher frequency images, thereby resulting in a more precise displacement estimate.

NOISE % ↓		SAMPLE WINDOW SIZE →		
		3x3	5x5	8x8
0	CORR	36.93	64.07	79.38
	SSD	66.67	78.26	82.03
5	CORR	30.53	60.30	78.86
	SSD	51.80	74.10	81.61
10	CORR	21.19	51.73	75.21
	SSD	30.44	60.33	76.67

Table 1: Comparing Laplacian-filtered correlation with Laplacian-filtered SSD.

The tests were conducted using the Mandrill Images described in [Glas83]. Gaussian noise of standard deviation 0, 5, and 10 percent of the intensity range were added to the second image which was translated by (3,5). The table entries are percentage of pixels with correct displacements. Matching was based on square windows of width 3, 5, and 8 pixels.

For large displacements, the use of band-pass filtered images and the coarse-fine strategy for matching is a natural generalization of the Laplacian-filtered matching technique. In this approach, at any given level of resolution the band-pass filtered image corresponds to Laplacian-filtering the low-pass filtered image that is faithfully representable (according to Nyquist criterion) at that resolution. The 3x3 search area used both by Burt et. al. and Glazer et. al. effectively limits the search to less than half the wavelength of the highest frequency information available at each level. This restriction also helps reduce the effect of false matches that may arise due to contrast variation between the images.

The band-pass filtered, coarse-fine search strategy tends to introduce some problems of its own. At occlusion boundaries, where there is a discontinuity in the displacement field, the coarse-resolution processing errors usually occur because 1) sampling windows will overlap across the boundaries and 2) the low-pass filtering process smooths the image across the boundaries. Since each pixel at a coarse level tends to cover a large area at the finest levels, these coarse-level errors tend to cause incorrect initial estimates to be used at the fine-level pixels, thus leading to a search in areas which do not include the correct match. Typically this creates a large area near the occlusion boundary with incorrect displacement field. Since these errors are primarily due to the hierarchical search strategy, it may be possible to eliminate some of these by using a single level search strategy. However, we believe a better approach would be to maintain the coarse-fine search strategy, but try to recognize such errors as they happen. Our confidence measure is, in fact, an attempt to do precisely that.

3. A CONFIDENCE MEASURE

As noted in the previous sections most correlation matching algorithms generate false matches in homogeneous areas, i.e., where there is a lack of any significant image structure, and around occlusion regions. Previous work that has attempted to provide smoothly varying dense displacement fields (Horn80, Glaz81, Nage83, Hild83) usually relies on the propagation of displacement information from image areas with significant intensity variations to homogeneous areas. However, when occlusion is also present, such estimates at the occlusion boundaries are usually incorrect due to occlusion effects and using them for initial estimates tends to confound the errors.

There have been efforts by Hannah [Hann74], Gennery [Genn80] and Burt [Burt83] to understand the reliability of correlation matching algorithms. Hannah observes that both the sharpness of the correlation surface at the point of best match and the similarity between the shape of the auto-correlation and the cross-correlation surfaces can be used to decide about the reliability of the match. However, she does not provide any concrete technique for using this

information. Gennery's measure requires a model of the camera noise and scaling effects between the images to be matched. This requires calibration of the camera set-up. Although this appears to be a robust measure, it is often the case that such calibration is difficult due to changes in illumination, surface reflectance, etc. We believe that it is possible to provide a confidence measure which does not depend on an a-priori model of the camera and the image noise. Burt provides a confidence measure, which in many ways is similar to our own. We describe Burt's measure in greater detail in section 3.3 and compare it with ours.

3.1 Properties of the SSD surface

We define an *SSD surface* as the surface formed by considering the Laplacian-filtered SSD values corresponding to different candidate displacements as the elevation at that displacement. This surface appears to contain a wealth of information about the nature of the image structures at the point being matched. Intuitively, it is clear that where there are significant intensity variations in the image, the match is likely to be reliable and unique, whereas at points in a homogeneous area, this is not so. This fact is noticeable in the shape of the SSD surface corresponding to such points. Usually, the SSD surface corresponding to a point with distinct image structure tends to have a sharp valley centered at the best match value, whereas at a homogeneous point the SSD surface is rather flat.

We conducted an empirical study of the behaviour *SSD surface*. For this study, we created a pair of synthetic images by digitally "cutting and pasting" pieces from two real images, photographed in our robotics lab [Elli84]. We then selected a number of specific points corresponding to typical image structures, (e.g., intensity corners, homogenous areas, occluded areas, etc.) and studied the behaviour of the Laplacian-filtered SSD surface at these points. The detailed results of this study are described in [Anan84]. Figures 3.1 and 3.2 show examples of such surfaces. Fig. 3.1 corresponds to an intensity corner which is visible in both images. Fig. 3.2 corresponds to an intensity corner in the first image which is occluded in the second image. These surfaces are inverted in the displays so as to enhance visibility. Thus the point of minimum SSD value is the peak in these two figures. We have marked the view-angle of the surface displays, as well as the minimum and maximum SSD values on the surfaces. In each figure, the point of minimum SSD value is marked with a "0" and the point corresponding to the correct displacement is marked with a "X". Note that Fig. 3.1 shows a distinct peak and the point of minimum SSD corresponds to the true-match point, whereas Fig. 3.2 shows erratic behaviour and the true-match point is away from the point of minimum SSD value.

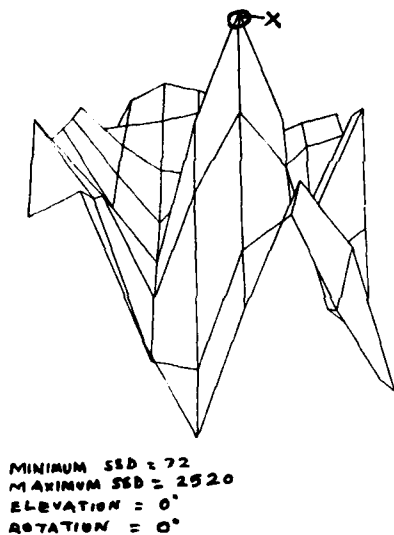


Figure 3.1: SSD surface at a corner

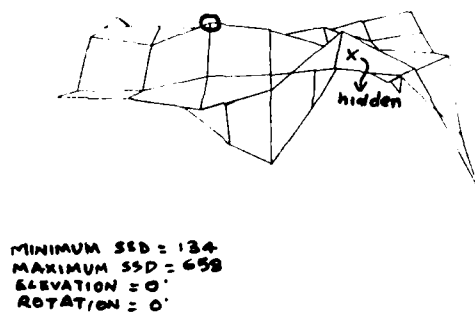


Figure 3.2: SSD surface at an occluded corner

The detailed study of these surfaces [Anan84] demonstrated how the SSD surface usually captures much of the information about the image structures as well as occlusion effects. Where a proper match exists (i.e., the non-occluded regions), the SSD value at the point of best match generally seems to be low. At occlusion areas this value is generally higher, and the selection of a proper match becomes difficult. We also found that the curvature of the SSD surface along different directions reflects the degree of variation in the image along those directions, and hence the uniqueness of the match estimate along that direction. These facts are combined in our confidence measure described in the next section.

3.2 The Confidence Measure

The behaviour of the SSD surface [Anan84] suggested that the confidence in the correctness of the match estimate should be directly proportional to the curvature of the SSD surface, and inversely proportional to the SSD value at the point of best match. Consider the normalized second derivatives of the SSD surface centered at the point of best match in the four directions 0, 45, 90 and 135 degrees. Each of these can be computed numerically using a 1×3 Laplacian operator oriented in the appropriate direction. We divide these curvatures by a weighted sum of the three SSD values used to compute the curvature. This is done both in order to normalize the curvature to be between 0 and 1, and to make it inversely proportional to the minimum SSD value.

In the formulae given below, the SSD surface is considered centered at the point of best match. The indexing is relative to that displacement, with index (0,0) referring to the displacement corresponding to the best match:

$$\begin{array}{ccc} S(-1, -1) & S(-1, 0) & S(-1, 1) \\ S(0, -1) & S(0, 0) & S(0, 1) \\ S(1, -1) & S(1, 0) & S(1, 1) \end{array}$$

We compute the four normalized directional second derivatives of the SSD surface as follows:

$$C0 = \frac{S(0, -1) - 2 * S(0, 0) + S(0, 1)}{S(0, -1) + 2 * S(0, 0) + S(0, 1)}$$

$$C45 = \frac{S(1, -1) - 2 * S(0, 0) + S(-1, 1)}{S(1, -1) + 2 * S(0, 0) + S(-1, 1)}$$

$$C90 = \frac{S(-1, 0) - 2 * S(0, 0) + S(1, 0)}{S(-1, 0) + 2 * S(0, 0) + S(1, 0)}$$

$$C135 = \frac{S(-1, -1) - 2 * S(0, 0) + S(1, 1)}{S(-1, -1) + 2 * S(0, 0) + S(1, 1)}$$

where $S(i, j)$ denotes the SSD value at position (i, j) relative to the point of best match.

At this point various possibilities arise. Since each of these four measures provide information specific to the corresponding direction they could all be separately maintained. Alternatively, a conservative measure will be to choose the minimum of these four measures. We have adopted this latter approach for our study. Hence our confidence measure is,

$$MIN(C0, C45, C90, C135)$$

3.3 The Confidence Measure of Burt, Yen and Xu

Burt, Yen and Xu [Burt83] describe a confidence measure which is very similar in form to the one describe here. Their measure uses the Laplacian-filtered correlation surface and takes the four directional second derivatives in a manner similar to ours. Their normalization technique is also similar to ours. However, the two measures differ in some significant ways.

First, our measure is based on the SSD surface, whereas their measure is based on the correlation surface. While the SSD function remains strictly positive, the correlation function can also have negative values. The presence of negative values can cause the normalized values to go below 0 as well as above 1. Although large negative values around the point of best match indicates a strong peak, in such situations, their confidence measure is below zero.

Second, in the strategy used by Burt, et al. confines the search at each level to a 3×3 area centered around zero displacement. This means that the curvature measures are taken at a point within this window, even though the actual displacement may be large. In band-pass filtered images, the structures can repeat themselves beyond a certain distance, thus causing false matches with high confidence (i.e., a unique match within the search window) to occur.

Finally, Burt's second derivative operators are centered at (0,0), even when the best match point occurs elsewhere within the 3×3 window. The 1×3 operator is too small to be a good approximation of the curvature at any point away from where it is centered. Therefore, if the displacement is not (0,0), Burt's measure may not be a good indication of the directional curvatures at the true match point,

3.4 A Demonstration

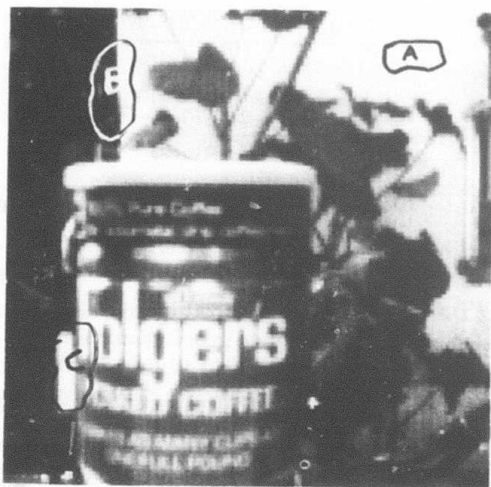


Figure 3.3: Folgers Image - first frame

Having defined our confidence measure, we now proceed to demonstrate its utility with an example. For this purpose, we chose a pair of real images, called *Folgers images*. The images are shown in figures 3.3 and 3.4 and constitute a stereo pair. There are two prominent surfaces at different depths, viz., the Folgers coffee can and the textured background with the plant. Occlusion at the left side of the can is clearly visible. This area in the first image has no matches in the second image. We used the Laplacian-filtered SSD as the match measure for computing the displacement field and the confidence measure. Figure 3.5a displays the displacement field and Figure 3.5b displays the confidence measure. The brightness of the confidence measure is proportional to the degree of confidence. The displacement field has been sub-sampled for convenience of display.

These figures reveal some important facts regarding the confidence measure. First, in large homogeneous areas of the image the confidence measure is low and the displacement estimates are often incorrect. For illustration purposes, we have drawn marked one such area as "A" in Figures 3.3, 3.5a, and 3.5b. Second, the confidence measure is also low along straight edges, although the component of the displacement vectors normal to the edge often seems to correct. One such area in the image is marked "B" in the three figures. Third, the confidence measure is low in occluded areas and around occlusion boundaries. One such area is marked "C" in the three figures. Finally, although the confidence measure is low both in homogeneous areas and occluded areas, it does not discriminate between the two.

In order to further illustrate the correlation between the confidence measure and the accuracy of the displacement estimates, Figure 3.6 displays only those displacement values which have a confidence of 0.3 or more.

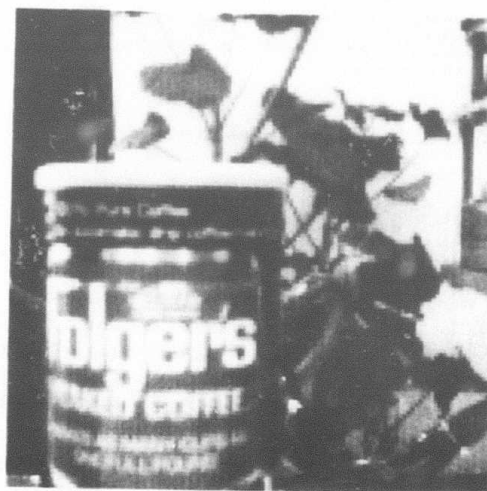


Figure 3.4: Folgers Image - second frame

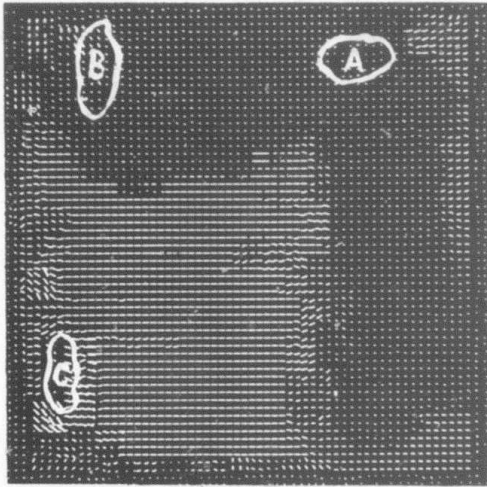


Figure 3.5a: The Folgers displacement field

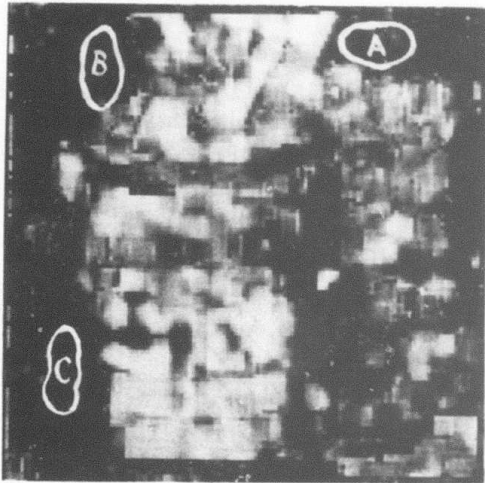


Figure 3.5b: The confidence measure for the Folgers pair

4. A MODIFIED SEARCH STRATEGY

The confidence measure which we described in the previous chapter is useful in isolating the areas in an image where the displacement estimates are unreliable, and often incorrect. In this chapter we describe two modifications to the search strategy of Glazer, Reynolds and Anandan which significantly reduce the errors in the displacement field, particularly near occlusion boundaries.

We describe briefly the search strategy of Glazer, et. al. in order to familiarize the reader with the terminology involved. The search begins at an appropriately coarse level so that all image displacements are less than one pixel distance at that level. For each pixel in the first frame at the coarse level (say level l), matches are found within a 3×3 window centered around the corresponding pixel in

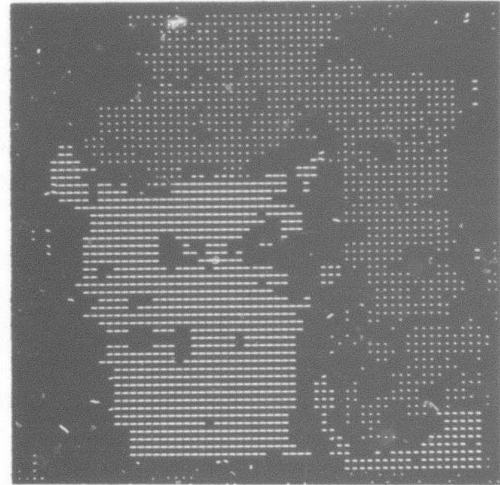


Figure 3.6: Folgers displacements with confidence greater than 0.3

This figure shows only the displacement vectors in Fig 3.5a which have a confidence measure greater than 0.3.

the second frame. Each of these estimates are then projected to the four pixels at level $l+1$ of the pyramid that are directly covered by each pixel at level l . The displacement estimates have to be multiplied by 2 in order to take into account the reduction in the pixel-width between levels l and $l+1$. For each pixel at level $l+1$, the search is conducted in a 3×3 area around these estimates. This process of projection and search is continued down to the finest level of the image pyramid.

Both modifications concern the projection of the displacement estimates from a coarse level to the next finer level. First, we restrict the projection of coarse estimates to only those with high confidence. Second, we allow the coarse level estimate at each pixel to be projected in an area larger than the 2×2 area directly covered by that pixel. Both these modifications are explained in greater detail in the following sections.

4.1 Restricting Projection to High Confidence Estimates

The restriction of projection to only high confidence estimates is suggested in Glazer, et. al. [Glaz83]. The motivation for this idea stems from the fact that when incorrect coarse-level estimates are projected down, the 3×3 searches at the finer levels are conducted in areas of the second frame that do not include the true-match point. This causes incorrect matches in all the subsequent levels. If on the other hand, these incorrect coarse-level estimates are altogether rejected, the finer-level searches can be conducted over larger area than the usual 3×3 windows, and the true-match can perhaps be recovered.

At any level l of the pyramid, the displacement updates at pixels where the confidence measure is low can be suppressed (this can be achieved by a simple threshold on the confidence measure). Wherever such updates are suppressed, we simply pass the displacement estimates from the parent pixel at level $l-1$ to the children pixels at level $l+1$. Assume that at such a pixel at level l , we are searching over a window of radius r (at level l), centered about the displacement estimate from level $l-1$. This corresponds to a window of radius $2 \times r$ at the next finer level. Hence the lack of update at level l would require that we search over a window of radius $2 \times r$ at the children pixels at level $l+1$. If there is still no update at this level the search window radius should be doubled at the next level below, and so on. This expanded search strategy is illustrated in Figure 4.1.

4.2 Modified Projection of Coarse Estimates

Our second modification to the search strategy involves the manner in which the coarse level displacement estimates are propagated to the fine level. The strategy of Glazer, et. al. projects the displacement value of a parent pixel as the estimate for all of its four children at the next finer level. At areas which are near discontinuities in the displacement field (e.g., occlusion boundaries), this approach can cause incorrect estimates to be projected from a coarse-level pixel to the finer level pixels. This occurs because, at a coarse-level of resolution the boundary of discontinuity can be placed only within a coarse accuracy. For fine-level pixels along one side of the boundary, it is then possible that the coarse estimates from the other side of the boundary are projected down. This causes the search at the subsequent levels to find incorrect matches.

We propose a slightly different method of projecting the coarse level displacement estimates to the next finer level. This idea is based on the "overlapping" pyramid idea of Burt, et. al. [Burt80]. Each coarse level pixel covers a 4×4 area at the next finer level, rather than the usual 2×2 . In this manner, each pixel at the finer level $l+1$ is considered to have four potential parents at the coarser level l (see Figure 4.2). We consider all the four estimates as possible initial estimates for the search at level l and conduct searches around each of these estimates. The displacement corresponding to the best match in this expanded search area is then chosen as the updated displacement estimate for that pixel at level $l+1$. In this way the pixels along the boundaries of displacement discontinuities are not bound to an incorrect coarse estimate. This allows for more precise placement of the boundaries of displacement discontinuities at the finer level.

It is obvious that we can combine both the modifications into a uniform algorithm. This would involve choosing the appropriate search radius for each of the estimates of the parent pixels according to their confidence value and their search radius at level l .

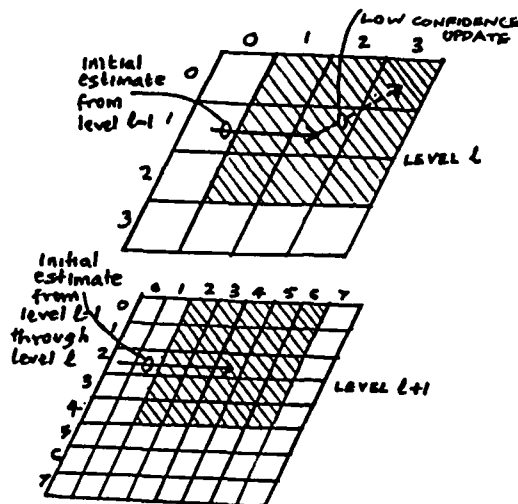


Figure 4.1: The expanded search area

Pixel (0,2) at level l has a low-confidence displacement update. Therefore, its initial estimate from level $l-1$ is passed down to its children pixels at level $l+1$. The expanded search area is shown cross-hatched.

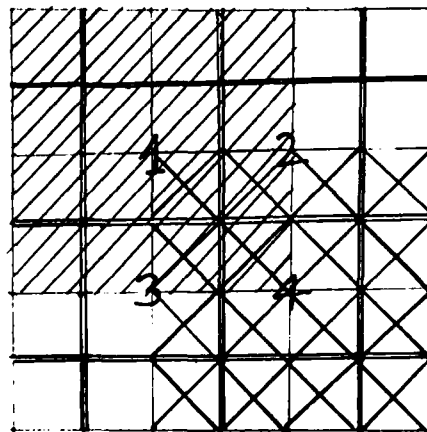


Figure 4.2: The overlapped pyramid projection

The thick double lines show pixel boundaries at level l . The thin lines show pixel boundaries at level $l+1$. The projection area of level l pixels 1 and 4 are shown in the figure. Note that each pixel at level $l+1$ has four parents at level l .

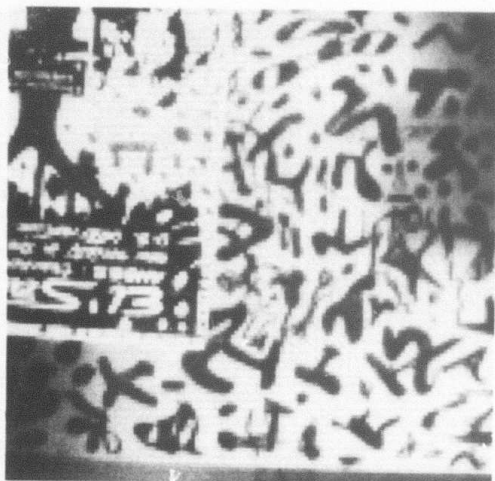


Figure 4.3: The Poster Image - first frame



Figure 4.4: The Poster Image - second frame

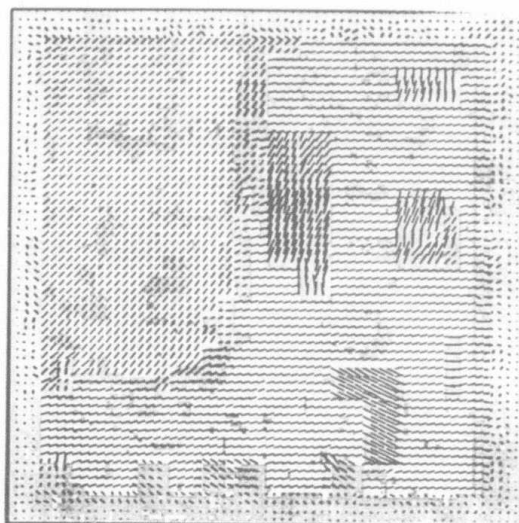


Figure 4.5: Results of the search strategy described in [Glas83]

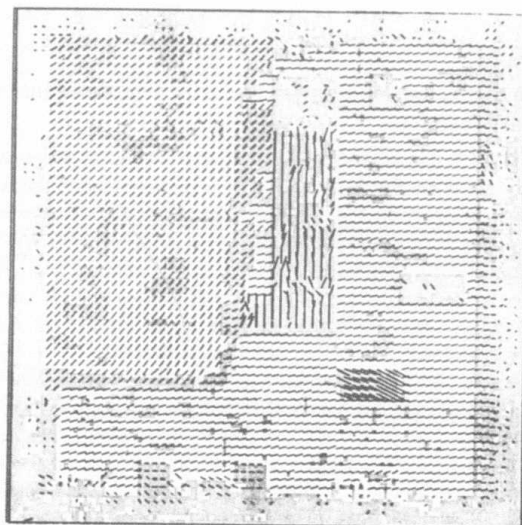


Figure 4.6: Results using restricted projection

In Figures 4.3 through 4.8, we demonstrate the effects of applying both these modifications to the search strategy. Fig 4.3 and 4.4 are two images from the sequence called *poster images*. The displacement estimates based on the old search strategy are shown in Fig 4.5. As in the case of our figures in Section 3, the displacement estimates have been subsampled to enhance visibility. The shaded areas have estimates with confidence below 0.3 and the white areas have estimates with confidence above 0.3. Note the predominance of the low confidence values around the occlusion boundary. Fig 4.6 displays the displacement estimates generated by a search strategy which incorporates only the first of the two modifications, viz., suppressing low confidence estimates. Again, the shaded areas are areas of low confidence (below 0.3). We make the following observations about Figures 4.5 and 4.6. First, the low-confidence areas

are rectangular in shape. This is due to the strict 1 to 4 projection used in these search strategies. In both cases if an error is made at a coarse level that is not suppressed, it is possible that the search areas at the finer levels for all the children pixels may no longer include the true match points. Since no information is used from the correct neighbours, these go uncorrected. Second, the restriction of projection of coarse estimates (Fig. 4.6) seems to improve the displacement estimates in some areas (of the background), whereas introduces more errors at others (near the occlusion boundary). This is because some of the low-confidence coarse estimates are actually correct. Eliminating these and

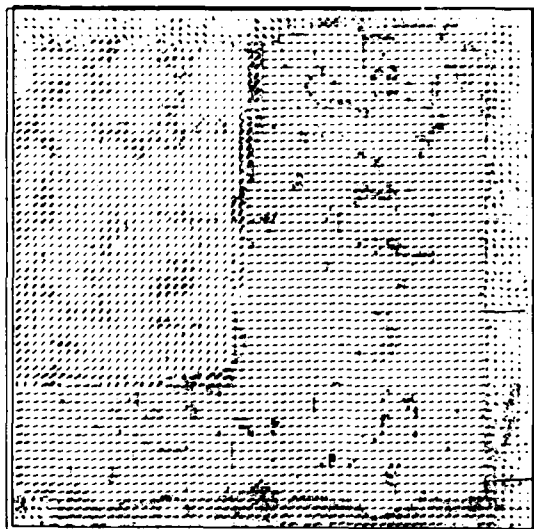


Figure 4.7: Results using overlapped projection

expanding the search area can lead to false matches due to repeated features.

Fig 4.7 displays the displacement estimates superimposed on low confidence areas as provided by a search strategy which incorporates only the second modification, viz., the overlapped pyramid projection. Fig 4.8 displays the displacement estimates superimposed on the shaded low confidence areas as provided by a search strategy that incorporates both the modifications mentioned above. In Figures 4.7 and 4.8, it is easy to see that a dramatic reduction in the size of the low-confidence areas has been achieved. Figure 4.7 shows that the modified projection strategy provides the major contribution to the improvement in the displacement field. This is so because, in this approach the coarse-estimates are not altogether eliminated. Instead, information is used from neighbours who may have correct estimates.

Later in this paper, we discuss other possible ways of utilizing the confidence measure to improve the matching results, all of which are currently under investigation.

5. APPLICATIONS AND FUTURE WORK

Thus far in this report we have described our confidence measure, and demonstrated its use in our modification to the search strategy. This measure can also serve as useful piece of information for techniques that process the flow field. In the following sections, we outline the immediate applications of the confidence measure, and describe the directions in which this study can be extended to include the various modifications necessary for other applications.

5.1 Use by Parameter Computation Algorithms

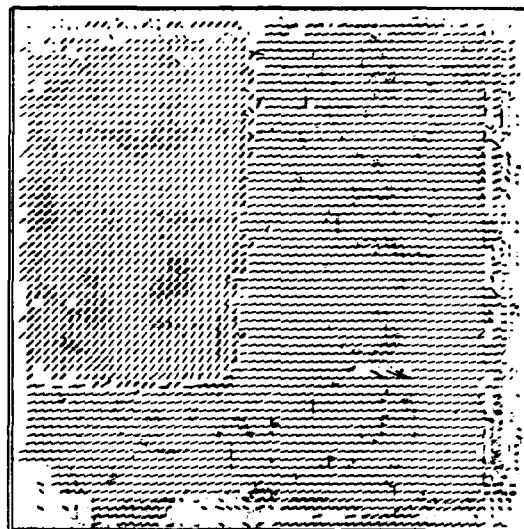


Figure 4.8: Results using both our modifications

One of the primary uses of image displacement fields has been to recover camera and object motion parameters and thereby the depth and the orientation of the image surfaces. Typically, techniques that address these problems involve solving a system of equations [Long81, Tsai84] or minimizing an error measure [Adiv84, Rieg84, Praz80, Lawt84].

These techniques can use the confidence measure in two ways. The first method is to eliminate the displacement estimates with low confidence measures. The second method is based on the observation that some of these techniques compute a global error or transform which has contributions from each displacement vector. This confidence can be weighted by the confidence measures. In this way less accurate displacement vectors (which typically have lower confidence), contribute less to the optimization process, thus enhancing the reliability of that process. As an example of this latter use, we point to Adiv [Adiv84], who attempts to segment the image into regions which have consistent displacement fields within those regions in order to compute the 3-d motion parameters corresponding to these fields. His technique is a multi-stage one, and involves transformations from the displacement vectors to affine-transformation parameter space, as well as least-square-error fits of 3-d transformations to the displacement fields. For both these purposes, he uses the confidence measure to weight the contribution of a displacement vector to this transformation and the error measure.

5.2 Future Work

Directional Information about Matching

The demonstrations of the behaviour of the SSD sur-

face at typical areas of the image with directional structures (e.g., edges) clearly showed that such directional information was indeed noticeable in the shape of the SSD surfaces. More specifically, we noted earlier that along edges in the images, we see a ridge like SSD surface where the orientation of the ridge corresponds to the orientation of the edge in the image. Hence, the directional confidence measure along the direction of the edge is low, whereas it is high in the direction perpendicular to it. It has also been well recognized by many researchers (e.g., [Glas81, Horn80]) that a directional feature in the image (say, an edge) can provide reliable information about the component of the displacements in the direction perpendicular to that feature, whereas it can provide no information about the component of the displacements along the direction of that feature.

There are two possible ways in which this directional confidence information can be utilized. The first is to use it in an algorithm similar to the modified search strategy described in section 4. In this case, the search area would not be expanded along the direction where the SSD surface shows significant variations. Instead, we can expand the search area in the direction along the ridge, thus obtaining a somewhat rectangular search area.

The second method is to use these in an algorithm that propagates information between neighbouring pixels especially along edges and curves in the image in order to bring together reliable information about different directions. One way of doing this is described in [Glas81]. Each pixel provides a linear constraint equation on the displacement vector at that pixel. In Glaser's approach, a least-square-error solution for the system of the constraint equations from neighbouring pixels along an edge is considered to be the true displacement vector for all pixels along the edge. Given that the SSD surface captures the directional information, we have available to us information similar to these constraint equations. An added benefit of using the SSD surface over the constraint equations is the fact that the SSD surfaces also provide information about noise variations and occlusion effects.

A more general way of using neighbour information for the improvement of the displacement estimates is to use relaxation-smoothing techniques. These are similar to the now classical Horn and Schunk smoothing technique [Horn80], or more closely to the constrained-smoothing approach described by Nagel [Nage83]. Nagel's approach integrates the intensity gradient constraint of Horn and Schunk with a smoothness constraint on the flow field which is somewhat different from theirs. The smoothness constraint usually involves the minimization of some function of the spatial-derivatives of the displacement field. Nagel modifies the function chosen by Horn and Schunk with weights that are inversely proportional to the intensity gradients at a given pixel. At each pixel, the component of the displacement vector along the direction of the intensity gradient is not allowed to vary significantly, whereas the component in the normal direction is allowed to change more freely.

Our study of the Laplacian-filtered SSD surface suggests that we can weight the function to be minimized using information from the SSD surface. Since Nagel's smoothness constraint uses image gradient information only from image, it is not sensitive to effects of noise and illumination in both images and to occlusion phenomena. The SSD surface includes such information and is therefore likely to prove more useful, particularly in scenes containing occlusion. One of our future goals is to pursue this approach and compare it with the approach of Nagel.

Recognition and Processing of Occlusion

Occlusion, although a source of failure and frustration for most algorithms that attempt to produce dense displacement fields, is very useful for the purposes of segmentation of the image into objects at different depth or with different movements. Therefore, any process that detects occlusion very early in the processing can be useful for focus of attention, tracking, and more accurate computation of the various image properties.

The confidence measure discussed in this study produces low values for both homogeneous areas as well as occlusion areas. However, often it may be possible to separate these situations using the information in the SSD surface. In real world images, it is usually the case that where there are occlusion boundaries, there are also discontinuities in the image texture. This suggests that all values on the cross-SSD surface will usually be high at occlusion boundaries, whereas they will be uniformly low at homogeneous areas. This observation can be useful in the identification of occlusion areas in an image. These issues will be the focus of our own future work in this direction.

REFERENCES

- [Adiv84] Adiv, Gilad, Determining 3-D Motion and Structure from Optical Flow Generated by Several Moving Objects, *in these proceedings*.
- [Agga81a] Aggarwal, J. K., Davis, L. S. and Martin, W. N., Correspondence Processes in Dynamic Scene Analysis, *Proceedings of the IEEE*, Vol. 69, Number 5, May 1981, pp. 562-572.
- [Anan84] Anandan, P. Computing Dense Displacement Fields With Confidence Measures in Scenes Containing Occlusion, *COINS Technical Report* University of Massachusetts, in press.
- [Bard80] Barnard, S. T. and Thompson, W. B., Disparity Analysis of Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, Number 4, July 1980, pp. 333-340.
- [Burt80] Burt, P. J., Hong, T. H. and Rosenfeld, A., Image Segmentation and Region Property Computation by Cooperative Hierarchical Computation, *IEEE Trans. Systems, Man, Cybernetics* 11, 1981, 802-809.

- [Burt82] Burt, P. J., Yen, C. and Xu, x., Local Correlation Measures for Motion Analysis: A Comparative Study, *IEEE Proceedings of PRIP*, 1982, pp. 269-274. Also *IPL-TR-024*, ECSE Dept., RPI, 1982.
- [Burt83] Burt, P. J., Yen C. and Xu X., Multi-Resolution Flow-Through Motion Analysis, *IEEE CVPR Conference Proceedings*, June 1983, pp. 246-252.
- [Elli84] Ellis, R. Robot Imaging System, *Laboratory for Perceptual Robotics* Internal Memorandum, no. 6, Jan, 1984, Univ. of Massachussets at Amherst.
- [Genn80] Gennery, D. B. Modelling the Environment of an Exploring Vehicle by means of Stereo Vision, *Report No. STAN-CS-80-805*, Dept. of Computer Science, Stanford University, California.
- [Gla81] Computing Optic Flow, Glazer F., *IJCAI-7*, Vancouver B. C., Canada, Aug. 1981, pp. 644-647.
- [Gla83] Glazer, F., Reynolds, G. and Anandan, P., Scene Matching by Hierarchical Correlation, *IEEE CVPR conference*, June 1983, pp. 432-441.
- [Hann74] Hannah, M. J. Computer Matching of Areas in Stereo Images, *Stanford A.I. Memo 239*, July 1974.
- [Hans80] Hanson, A. and Riseman, E. M., Processing Cones: A Computational Structure for Image Analysis, in *Structured Computer Vision*, (Tanimoto, S. and Klönger, A., editors), Academic Press, New York, 1980.
- [Hild83] Hildreth, E. C. Computing Velocity Field Along Contours, *ACM Siggraph/Sigart Workshop on Motion* Toronto, Canada, Apr. 1983, pp. 26-32.
- [Horn80] Horn, B. K. P. and Shunck, B. G. Determining Optical Flow, MIT A.I. Memo Number 572, April 1980.
- [Lawt84] Lawton D. T. Processing Dynamic Image Sequences From A Moving Sensor, *Ph.D Dissertation and COINS Technical Report 84-05*, University of Massachussets, 1984.
- [Long81] Longuet-Higgins, H. C., A Computer Algorithm for Reconstructing a Scene from Two Projections, *Nature* 293, 1981
- [Mora81] Moravec, H. P. Robot Rover Visual Navigation, UMI Research Press, Ann Arbor, Michigan, 1981.
- [Nage83] Nagel, H.-H., Constraints for the Estimation of Displacement Vector Fields from Image Sequences, *IJCAI-83*, Karlsruhe, West Germany, pp. 945-951.
- [Praz80] Prazdny, K., Egomotion and Relative Depth Map from Optical Flow, *Biology and Cybernetics*, Vol. 36, 1980, pp. 87-102.
- [Wong78] Wong, R. Y. and Hall E. L. Sequential Hierarchical Scene Matching, *IEEE Transactions on Computer*, Volume 27, No. 4, 1978, pp. 359-366.

Map-Guided Feature Extraction From Aerial Imagery

David M. McKeown, Jr.
Jerry L. Denlinger
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213*

Abstract

In this paper we discuss the use of map descriptions to guide the extraction of man-made and natural features from aerial imagery. An approach to image analysis using a region-based segmentation system is described. This segmentation system has been used to search a database of images that are in correspondence with a geodetic map to find occurrences of known buildings, roads, and natural features. The map predicts the approximate appearance and position of a feature in an image. The map also predicts the area of uncertainty caused by errors in the image to map correspondence. The segmentation process then searches for image regions that satisfy 2-dimensional shape and intensity criteria. If no initial region is found, the process attempts to merge together those regions that may satisfy these criteria. Several detailed examples of the segmentation process are given.

1. Introduction

This paper describes MACHINESEG, a program that performs map-guided image segmentation. It uses map knowledge to control and guide the extraction of man-made and natural features from aerial imagery using region-growing techniques. We use the CONCEPTMAP database¹ from the MAPS system² as our source of map knowledge. In the CONCEPTMAP database, map knowledge is represented as three dimensional descriptions of man-made features, natural features, and conceptual features. Examples of man-made features are buildings, roads, and bridges; natural features are rivers, lakes, and forests, and conceptual features are political boundaries, residential neighborhoods, and business areas. These feature positions are represented in the map database in terms of $\langle \text{latitude}, \text{longitude}, \text{elevation} \rangle$. In this paper we will discuss the extraction of man-made and natural features.

*This research was sponsored by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-81-K-1539. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

In this paper, we refer to *regions* as areas of more-or-less uniform pixel intensity which may or may not have interpretations as real world surfaces or objects. *Features* are regions that have been recognized and interpreted by a program, or have been outlined by a human, or can be characterized by a simple set of position, shape, size, and spectral properties. *Image segmentation* is the process of generating candidate regions for the feature extraction process. *Feature extraction* is the recognition of a region with particular properties by application of one or more tests. The result of feature extraction is the generation of a set of regions in the image which satisfy feature extraction criteria specified by a user or high-level process. In MACHINESEG we have combined segmentation and labeling so that the segmentation process presents candidates for evaluation to the labeling process. Once a region is identified as a feature, special evaluation procedures are used.

2. Map-Guided Segmentation

The notion of map-guided image segmentation is not a new one. Many researchers have discussed the use of *a priori* knowledge of various object features such as size, shape, orientation, and color to extract and identify features from an image. However, there are few, if any, examples of systems that can systematically search through a database of images looking for examples of particular objects or classes of objects. In this paper we present one such system which uses constraints derived from a map database to perform segmentation in aerial imagery.

It is important to characterize what we mean by "map-guided" image segmentation.

Map-guided image segmentation is the application of task-independent spatial knowledge to the analysis of a particular image using an explicit map-to-image correspondence derived from camera and terrain models.

Map-guided segmentation is not interactive editing or computation of descriptions in the image domain, since these descriptions are valid only for one specific image. In MAPS there is a geodetic description

(*latitude, longitude, elevation*) for each map entity in the CONCEPTMAP database. This description is in terms of *points*, *lines*, and *polygons*, or collections of these primitives. Features such as buildings, bridges, and roads have additional attributes describing their elevation above the local terrain, as well as their composition and appearance. The location of each map feature in the database can be projected onto a new image using a map-to-image correspondence maintained by MAPS. Likewise, a new map feature can be projected onto the existing image database. If camera model errors are known, one can directly calculate an uncertainty for image search windows. Further, as new features are acquired their positions can be directly integrated into the map database.

Figure 1 gives a schematic description of the map-guided feature extraction process in MAPS. There are two basic methods for applying map knowledge to the extraction of features from aerial imagery. The first method uses generic knowledge about the shape, composition and spectral properties of man-made and natural features. The second uses map-based template descriptions. These descriptions are stored in the CONCEPTMAP database and represent knowledge about known buildings, roads, bridges, etc. This knowledge includes geodetic position, shape, elevation, composition and spectral properties. In the second case, the position, orientation, and scale are constrained whereas in the first, only the scale can be determined. In both cases, in

order to operationalize spatial knowledge for the analysis of a particular image, a map-to-image correspondence is performed. In this paper we discuss the implementation and performance of a region-based segmentation/feature extraction program which has been integrated to use these map constraints to guide segmentation.

Application areas for this approach include digital mapping, remote sensing, and situation assessment. More specifically, tasks that can capitalize on *a priori* map knowledge to constrain 'where to look' and 'what to look for' may provide sufficient context for inherently weak methods of feature extraction to be effective. Rather than looking for "perfect" segmentations, our approach extracts segments characterized as "islands of reliability" for some particular instance or class of object. These local regions can be further analyzed by modules that bring to bear more task-specific or object-specific knowledge to confirm or refute the initial analysis.

In Section 3 we discuss the organization of MACHINSEG and describe various constraints that can be applied during region-growing. In Section 4 several examples illustrate the capabilities of map-guided segmentation using map-based template descriptions and generic feature descriptions.

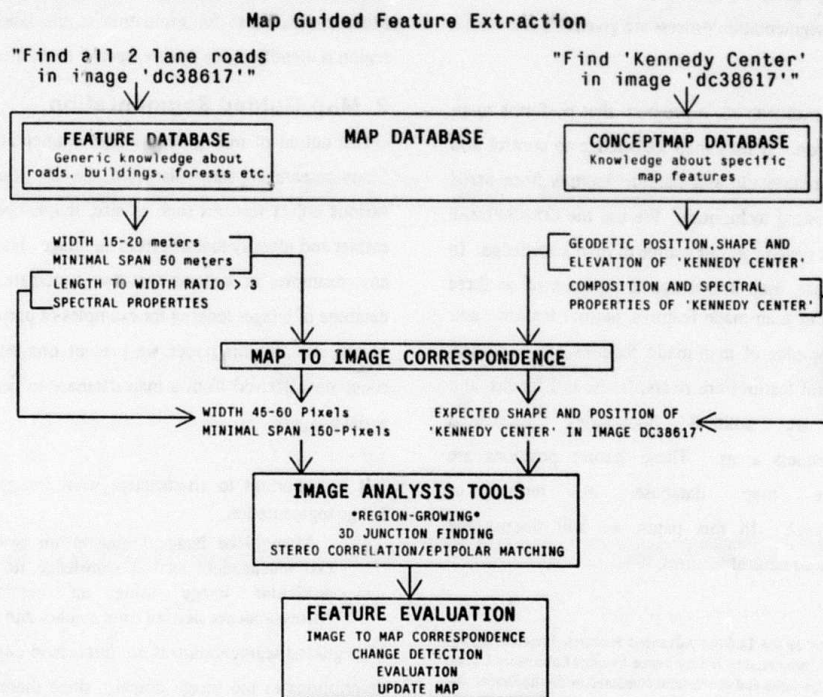


Figure 1: Map-Guided Feature Extraction

3. MACHINESEG: Region-Based Segmentation Using Constraints

In this section we describe the implementation and control of a region-based image segmentation program that utilizes shape and spectral constraints to control the merging and selection of primitive regions. These constraints can be specified by an interactive user, or by another program.

3.1. Region Growing as Segmentation

Region growing is a well understood technique in image processing and computer vision for providing a region-based segmentation of an image. Ballard and Brown³ provide a good introduction and Yakimovsky⁴ gives a detailed treatment.

One problem with region growing is that it is difficult to know when to stop merging regions together. Standard techniques involve thresholds on edge strength between regions and/or on merge compatibility based on spectral similarity. These thresholds may be difficult to select, especially if one requires robust behavior over many different images at a variety of resolutions. If we merge until there is only one region left, we have obviously gone too far. If we stop the process based on counting merge iterations or other *ad hoc* considerations, some regions may have merged into a stable state, while others will still be in several pieces or may have already been merged into the background. This problem is caused by the fact that semantically meaningful features will not necessarily have good edges. The underlying assumption is that regions of (nearly) homogeneous intensity in the image correspond to objects or surfaces which are physically realized in the scene. However, as we well know, some regions will have weak edges, because they do not differ much from the background, and edges can exist where there is really no boundary between objects. Shadows, highlights and occlusions also violate this assumption and complicate the processing of aerial imagery.

We therefore introduce a method for stopping the merging of *specific regions* rather than trying to determine when the entire image segmentation should be terminated. The approach we have taken is different from classical segmentation in that we do not necessarily break up the image into disjoint regions so that each pixel is part of some region. Rather, we have developed a method for finding features that meet some specific criteria. By changing the selection criteria it is possible to assign more than one region interpretation to a pixel in different executions of the region-merging process. Criteria can be used to look for features whose exact shape is unknown but can still be characterized generically. For example, if we want to find roads we can look for linear features. If we are segmenting an airport scene, we

may find large grassy areas between runways by looking for blob features of some minimum size. To find more complex shapes, such as specific buildings, we use template matching. In the following section we discuss the merging algorithm that allows us to search for linear regions, compact regions, blob regions, or to match regions to a shape template.

3.2. The Algorithm

The basic procedure for producing regions which satisfy a particular set of criteria is as follows:

1. Interactive user or program invokes MACHINESEG with an image name and sub-image area.
2. The sub-image area is smoothed using edge-preserving smoothing⁵.
3. Edge extraction is performed, and seed regions, primitive 8-connected regions of homogeneous image intensity, are produced.
4. A "state" file containing the names of the intermediate images, edge and region data structures are saved. We make use of this file to restore the initial state of primitive regions when changing criteria. This is discussed in Section 3.5.
5. Match criteria are selected using map-based generic or template descriptions as described in Figure 1.
6. Regions are merged based on the strength of the edges between regions. Resulting merged regions are evaluated and marked for special handling if they satisfy the criteria.

We store a list of the edges between regions, sorted by the strength of the edge. We simply scan down the list of edges, starting with the weakest, and merge the two regions that share that edge. Each time a new region is created by merging two other regions, the new region is "scored" against the specified set of area, intensity, and shape criteria to determine if it is similar to the prototype region we are looking for. If it is similar enough, we mark the merged region. Users can specify that after a region has been marked, it will not be merged any further unless the resulting region would improve the criteria "score" or, alternatively, if it would simply meet the criteria. Meeting the criteria allows newly merged regions to get locally worse scores in order to permit future merge operations. If the resulting merge must improve the criteria, the region score must monotonically increase with each merge. Various high-level strategies may select the appropriate evaluation method. For example, in template matching, we require that merges improve the score since this helps prevent small appendages from being merged with the feature. In looking for linear features, we perform any merge as long as the resulting feature would still meet the criteria.

The underlying idea behind our region merging scheme is that, if a feature exists with the characteristics we are looking for, a significant portion of that feature will eventually be merged into a single region.

We then stop the merging of that feature with other regions if the merge would not maintain or improve the region model. We make the usual assumptions that the features we are trying to extract will have good edges. As long as the edges between the object and the background material are stronger than the edges between the subregions of the object itself, this method will work reasonably well. If the edges between the background and the region are weak (ie. the average intensities do not differ by much) this technique will not perform better than classical techniques. Figure 2 shows the region merge evaluation loop performed by MACHINESEG.

3.3. Evaluation Criteria

Different criteria can be set by a user or by some high-level process, such as the map database, to determine what types of regions to search for. We can look for regions within a certain range of average intensity, area, compactness, linearity, or by matching regions with a specific 2D shape. For example, when searching for tarmac regions in airport scenes we use combinations of these criteria. In the following sections we discuss the currently implemented selection criteria.

3.3.1. Average Intensity

Average intensity of the region is the weakest constraint. We have mainly applied this criteria to identify possible shadow regions using an analysis of the image intensity histogram to specify criteria. To select regions of a specific intensity, the user/process specifies a range in which the average intensity of the region must fall.

3.3.2. Area

Area is simply the number of pixels in the region. The area measure can be used by itself to find background areas which often appear as large, homogeneous regions immediately after the seed regions are grown. The area criteria is most often combined with other metrics such as linearity and compactness to avoid computation on regions that are too small or large. To select regions of a specific area, the user/process specifies a range of acceptable region areas.

3.3.3. Compactness

The compactness of a region is defined by

$$compactness = \frac{4\pi \times area}{perimeter^2}$$

By using a low compactness, we can find blob features, or features that are roughly circular. Features with high compactness are candidates for man-made structures. To limit regions using compactness, the user/process specifies a compactness range which is acceptable.

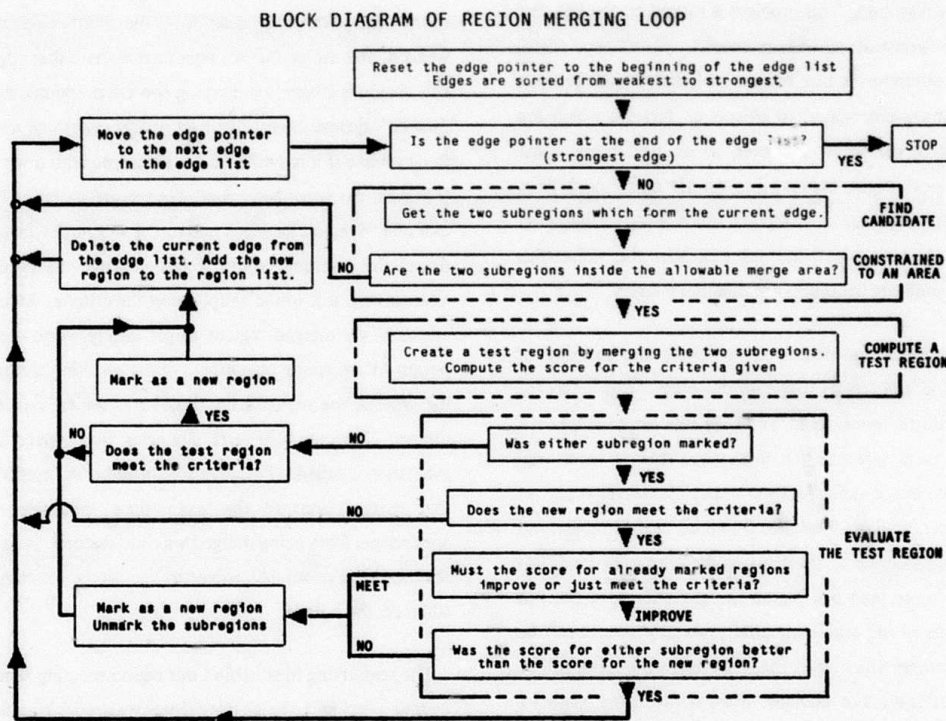


Figure 2: Merge and Criteria Evaluation in MACHINESEG

3.3.4. Linearity

The linearity measure is an heuristic designed to give high values for long, narrow features and lower values for other shapes. For rectangular features, the linearity is approximately equal to the length-to-width ratio, independent of orientation. Thus, this measure can not only detect linear features, but also gives some measure of how linear the feature is. To use the linearity criterion, a user specifies a minimum linearity. Regions with a linearity greater than or equal to the value specified are then classified as being linear.

We use the length and width of the bounding box of the region, its area and perimeter to compute the linearity measure. If the region is a narrow rectangle, it will lie diagonally in its bounding box and its length will be approximately

$$length = \sqrt{MBRH^2 + MBRW^2}$$

where $MBRH$ and $MBRW$ are the height and width of the bounding box. Still assuming the region is a rectangle, we can compute its width as

$$width = \frac{area}{length}$$

The length-to-width ratio is therefore

$$\begin{aligned} \frac{length}{width} &= \frac{length}{area / length} = \frac{length^2}{area} \\ &= \frac{MBRH^2 + MBRW^2}{area} \end{aligned}$$

We use either this expression or its reciprocal, whichever is larger. This formula will give the length-to-width ratio for regions that are rectangular. However, for regions that are not rectangular, the result in this form is meaningless. By adding a further dependence on perimeter, we can reduce the score for regions that have appendages. Since the formula is designed to give high values for rectangles, a perimeter value different from that which would be expected for a rectangle should decrease the score. That is, we will add a dependence on perimeter in such a manner as to decrease the value of this formula for non-rectangular regions. The desired effect can be achieved by multiplying by a correction factor.

$$correction\ factor = \frac{2 \times (width + length)}{perimeter}$$

Note that this is a unitless quantity. The value of this expression will be approximately 1.0 for a rectangle but will decrease with imperfections. The expression will not be exactly 1.0 since we use approximate *length* and *width* as computed above. For some shapes, (circles, for example) the value of this expression can be greater than 1.0. Since this can only occur when the region is fairly compact, and compact regions are not linear, we multiply by the reciprocal of this expression if it is greater than 1.0. The square of this expression seems

to give better results in practice since this further increases the sensitivity to imperfections -- this also eliminates the need to compute square roots when the entire expression for linearity is expanded. Thus, for regions that are approximately rectangular, we compute the length-to-width ratio. For other regions, the score computed for linearity is relatively low.

3.3.5. Template Matching

Template matching can be used to look for a region having a specific shape. The measure computed is the percentage of overlap between the region being measured and the template shape. The template shape, given in polygon form, and the minimum percentage of overlap must be specified. The shape may be specified either interactively or from a stored database file. The template shape is scan-converted into a matrix to simplify the shape comparison process. Scan-conversion of the regions is not necessary since they are stored in image format as a part of the region growing process. To compute overlap, we find the centroids of both regions and shift the region matrix so that the centroids line up. Overlap is defined as the total number of pixels matched from both regions (ie. twice the number of overlapping pixels), divided by the sum of the areas of the two regions.

$$overlap = \frac{2 \times intersection}{area1 + area2}$$

where *intersection* is the area of the intersection and *area1* and *area2* are the areas of the two regions under comparison. This gives an overlap of 1.0 for identical regions. It also gives low overlaps for regions whose size is very different, even if one of the regions is wholly contained in the other. For regions of the same size, it will give scores in proportion to the area of intersection.

This comparison can be performed at an arbitrary number of orientations spaced at equal intervals; in some cases (eg. template criteria) we know the orientation approximately and need only one orientation. In other cases, we may have a good model of the expected shape, but have weak constraints on its orientation. For multiple orientations, we compute the overlap for all orientations and use the maximum value. This comparison is obviously computationally expensive, but many regions may be excluded from this operation simply because their area is too large or too small for a match of the desired percentage to be possible. For example, if we are looking for an 80 percent overlap with a feature containing 100 pixels, we only need to perform the overlap computation for regions with areas between 67 and 150 pixels. The performance of the overlap computation could be improved using alternative formats such as run-coding, or a variation of chamfer matching. However, in the first case this would require additional storage and the computation of new run-codes for merged regions. Additionally, our method allows for holes

within the template region, which would complicate the straightforward run-code algorithm as well as chamfer matching as implemented by a grassfire algorithm.

3.4. Limiting the Search Area

In addition to providing the ability to look for regions of specific shape, other actions of the region grower can be controlled by higher level processes. The region merging can be limited to specific image sub-areas to improve efficiency. This might be done by a high level process whose goal was to complete the merging in a certain area to determine if a feature was present. This may be useful in analysis of other areas of the image if some specific information is known about the scene being segmented. For sub-area merging, the edge list is traversed as usual, but merging of regions is disallowed if neither region is wholly contained in the sub-area. Since the region merging is expensive, limiting the search area can achieve significant speed-up.

3.5. Suspension of Merging

Another form of high-level control is the ability to stop region merging at a specified point and return control to a higher level. This can be done when:

- Some number of regions that fit the feature criteria are found.
- A particular marked feature is updated as having been extended.
- A certain number of merge cycles have been performed.

This gives a high-level program a fine grain of control over the segmentation process as well as the ability to modify the criteria or search in a small area. After analysis of the results of an initial region merge, criteria can be relaxed or made more restrictive, based on the goals of segmenter. Merging may be restarted from the initial seed regions, or resumed from the point of suspension. This flexibility allows us to implement high-level strategies such as best-first or bottom-up propagation of weak hypotheses. Similar control over parameters by evaluation procedures are described by Selfridge⁶.

4. Some Examples

The following examples illustrate map-guided segmentation using the MACHINESEG program. The first three examples show the extraction of buildings and natural features from images of the Washington D.C. area using the CONCEPTMAP database. The final two examples show the use of map-derived size and shape criteria to find instances of generic objects in an image of National Airport.

4.1. Map Guided Template Matching

The following three examples were generated using the CONCEPTMAP program. This program allows a user to specify a feature in the database and an image in which to look for the feature. The program then creates a template feature using the map description in

the database and the map correspondence for the image given. The template feature description determines both the area to segment in and the shape to look for. CONCEPTMAP invokes the MACHINESEG program to find a feature of a specific shape within a small context area of the image. The regions shown in Figures 3,4, and 5 were extracted using a match score of 0.8 (eighty percent). The context area was approximately twice the size of the predicted feature. Using a small area helps to reduce false alarms from similarly shaped features in the same area. This is usually only a problem in lower resolution images.

Figure 3 shows the results of processing for the feature Kennedy Center in five different images. Image patches labeled DC1013 and DC1109 are digitized from aerial photographs taken at scale 1:60000, DC1420 was taken at scale 1:36000, and DC38618 and DC38617* were taken at scale 1:12000. In these scales, one pixel is about equal to 5, 3 and 1 meters square, respectively. The image labeled DC38617* had been segmented by hand to create the database feature used for matching. In the lower resolution images, the contrast is rather poor, but large portions of the feature were still detected. In the higher resolution images, the roof of the Kennedy Center is not homogeneous. In these images, the feature is not merged together into a single region that matches the shape specified until fairly late in the merging process. The tail on the feature in DC38618 is a piece of sidewalk that was merged into part of the building before the feature was merged together.

Figure 4 shows the results of processing on the feature Executive Office Building in four different images. One of the images of the feature is shown on the left with the segmentation result overlayed and appearing as a dark outline. On the right are the outlines of the predicted feature shapes, the extracted features, and the superpositions of the predicted and extracted features, showing their relative positions in the image. Note the recovery from a significant correspondence error in one of the examples. The resolution for each image is given on the far right.

Figure 5 shows the results of processing for the feature McMillan Reservoir in four different images. One image of the feature is shown on the left with its segmentation result overlayed and appearing as a dark outline. In this image, part of the feature is not visible since the feature is on the edge of the image and is clipped. When this happens, the map-to-image correspondence of the database feature onto the image results in a template feature clipped to the image bounds. The resulting shape is approximately the same shape as that in the image to be segmented. The accuracy of locating the partial feature is usually

the same as for location of the whole feature. On the right and bottom of figure 5 are the outlines of the predicted feature shapes for the other three images along with the extracted features, and the superpositions of the predicted and extracted features, showing their relative positions in the image. The superposition of the predicted and extracted shape is not shown for the example at the bottom due to space limitations. The region at the bottom of the figure was also on the edge of the image except in this case almost all of the feature was off of the image. The resolution for each image is given on the far right.

4.2. Using Generic Descriptions

In addition to the use of specific map feature templates, MACHINESEG can be used to find regions having generic shape or spectral properties. Figure 6 is a photograph of the terminal building area at National Airport, Washington D.C.. We have been using MACHINESEG to provide region candidates to a rule-based system for photo-interpretation, SPAM⁷. Figures 7 and 8 show line drawings of the regions extracted from Figure 6. The criteria for Figure 7 were established to produce large blob regions, which might correspond to tarmac, grassy areas between runways, or parking lots. A histogram of initial seed region areas was used to select an area criteria based on the distribution of large initial regions. Since we were searching for blob regions, a compactness criteria which excluded compact regions was selected.

The interaction between SPAM and MACHINESEG is an example of a high-level process generating tasks for low-level image processing. Since SPAM maintains models of its current view of the state of the airport interpretation, it can predict image sub-areas within which particular features may be found and shape criteria for those features. For example, when a long linear region is produced by MACHINESEG, several hypotheses (interpretations) may be produced, such as runway, taxiway, access road, shadow region, etc. In order to verify these hypotheses, SPAM may invoke MACHINESEG to attempt to extend a linear region at either of its ends. Since the location of the feature endpoints, width, and other shape attributes are known, criteria can be specified which constrain region merging to an image sub-area looking for new regions with similar properties.

Thus, Figure 8 shows the results of MACHINESEG segmentation using linear and small area criteria. The number of linear regions produced can be controlled by increasing the linearity criteria to make it more selective. While this segmentation is not perfect, it does give a good set of candidate regions for high level processing. The length, width and area ranges can be specified using ground distances (ie. meters, feet)

and transformed automatically into pixel distances using map-to-image correspondence. Current generic feature criteria include runways, taxiways, access roads, parking lots, grassy areas, tarmac, hangars, and terminal buildings.

5. Conclusion

In this paper we describe MACHINESEG, a program that performs map-guided image segmentation. The use of shape and spectral criteria to control merging of regions within a region-growing paradigm is discussed. Examples of the use of a feature description from a map database to guide feature segmentation from an image database using explicit map-to-image correspondence are presented. The use of generic map-based descriptions of shape find instances of classes of objects is presented. This program has been integrated into the MAPS system and uses the CONCEPTMAP database as a source of feature descriptions. It is also used as a component of a rule-based system (SPAM) for photo-interpretation.

6. Acknowledgements

David Smith implemented several novel aspects of the low-level region grower, particularly, an efficient representation of region adjacency and edge strength. Steve Shafer, Takeo Kanade, and Victor Milenkovic commented on an earlier version of this paper. Kim Faught aided in the preparation of this paper.

7. Bibliography

1. McKeown, D.M., "Concept Maps," *Proceedings: DARPA Image Understanding Workshop*, Sept. 1982, pp. 142-153, Also available as Technical Report CMU-CS-83-117
2. McKeown, D.M., "MAPS: The Organization of a Spatial Database System Using Imagery, Terrain, and Map Data," *Proceedings: DARPA Image Understanding Workshop*, June 1983, pp. 105-127, Also available as Technical Report CMU-CS-83-136
3. Ballard, D. H. and Brown, C. M., *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
4. Yoram Yakimovsky, "Boundary and Object Detection in Real World Images," *Journal of the ACM*, Vol. 23, No. 4, 1976, .
5. Nagao, M., Matsuyama, T., "Edge Preserving Smoothing," *Proceedings of the Fourth International Joint Conference on Pattern Recognition, IJCPR*, Kyoto, Japan, November 1978, pp. 518-520.
6. Selfridge, P. G., "Reasoning About Success and Failure in Aerial Image Understanding," Tech. report 103, University of Rochester, May 1982, Ph.D Thesis
7. McKeown, D.M., and McDermott, J., "Toward Expert Systems for Photo Interpretation," *IEEE Trends and Applications '83*, May 1983, pp. 33-39.

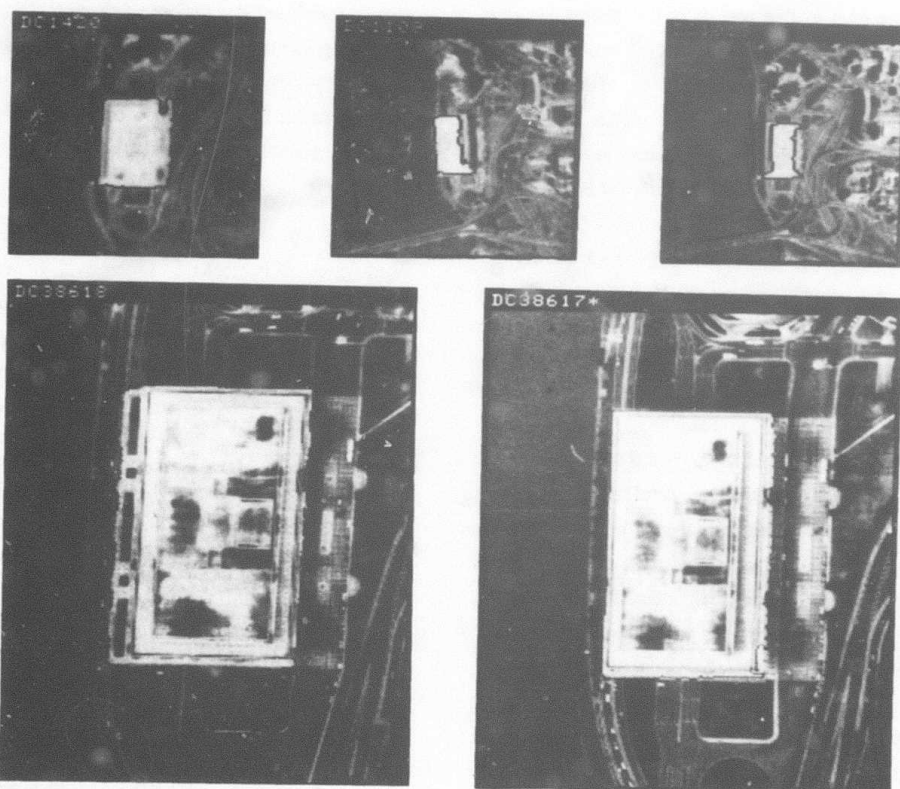


Figure 3. Kennedy Center

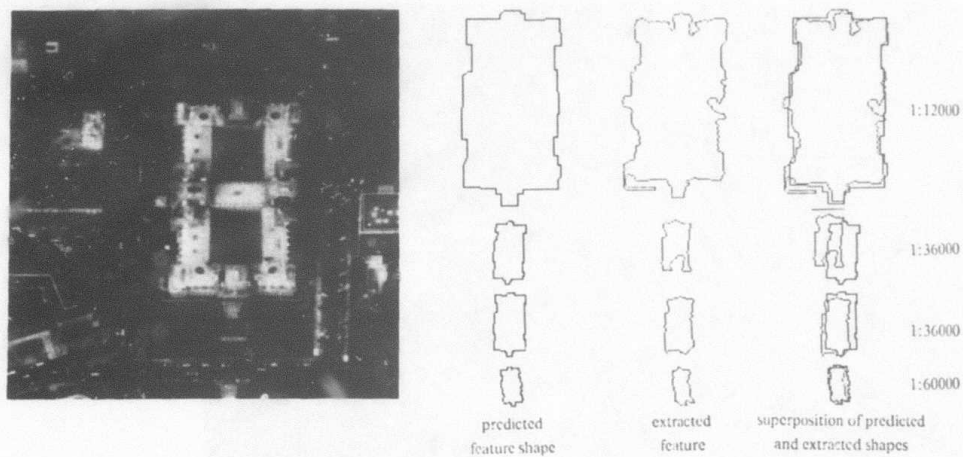


Figure 4: Executive Office Building

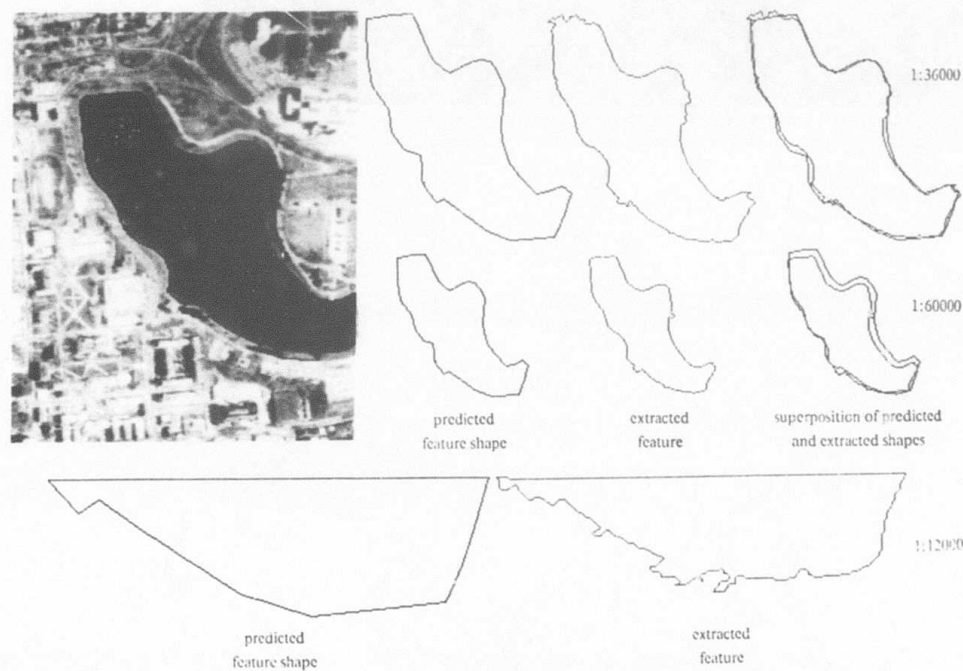


Figure 5: McMillan Reservoir



Figure 6: Terminal Building Area: National Airport

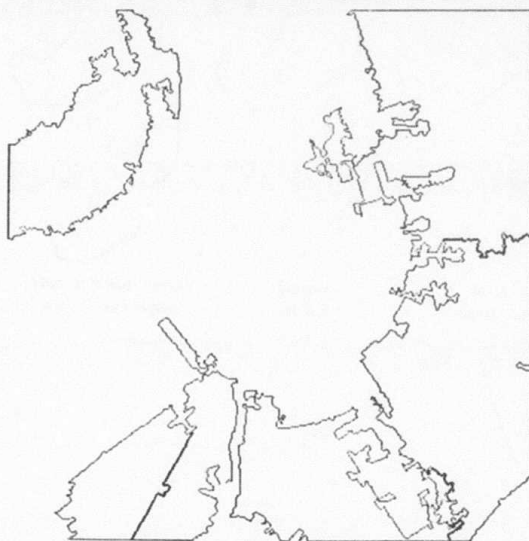


Figure 7: BI OB Regions Extracted



Figure 8: LINEAR Regions Extracted

ILL-POSED PROBLEMS AND REGULARIZATION ANALYSIS IN EARLY VISION

Tomaso Poggio and Vincent Torre

Massachusetts Institute of Technology, Cambridge, Mass.

Abstract

One of the best definitions of early vision is that it is inverse optics — a set of computational problems that both machines and biological organisms have to solve. While in classical optics the problem is to determine the images of physical objects, vision is confronted with the inverse problem of recovering three-dimensional shape from the light distribution in the image. Most processes of early vision such as stereomatching, computation of motion and all the "structure from" processes can be regarded as solutions to inverse problems. This common characteristic of early vision can be formalized: *most early vision problems are "ill-posed problems" in the sense of Hadamard.* We will show that a mathematical theory developed for regularizing ill-posed problems leads in a natural way to the solution of early vision problems in terms of variational principles of a certain class. This is a new theoretical framework for some of the variational solutions already obtained in the analysis of early vision processes. It also shows how several other problems in early vision can be approached and solved.

Variational Solutions to Vision Problems

In recent years, the computational approach to vision has begun to shed some light on several specific problems. One of the recurring themes of this theoretical analysis is the identification of physical constraints that make a given computational problem determined and solvable. Some of the early and most successful examples are the analyses of stereomatching (Marr and Poggio, 1976, 1979; Grimson, 1981a,b; Mayhew and Frisby, 1981; Kass, 1984; for a review see Nishihara and Poggio, 1984) and structure from motion (Ullman, 1979). More recently, variational principles have been used to introduce specific physical constraints. For instance, visual surface interpolation can be derived from the minimization of functionals that embed a generic constraint of smoothness (Grimson, 1981b, 1982; Terzopoulos, 1983, 1984a). Computation of visual motion can be successfully performed by finding the smoothest velocity field consistent with the data (Horn and Schunck, 1981; Hildreth, 1984a,b) and shape can be recovered from shading information in terms of a variational method (Ikeuchi and Horn, 1981). The computation of subjective contours (Ullman, 1976; Brady et al., 1980; Horn, 1981), of lightness (Horn, 1974) and of shape from contours (Barrow and Tennenbaum, 1981; Brady and Yuille, 1984) can also be formulated in terms of variational principles. Terzopoulos (1984a, 1985) has recently reviewed the use of a certain class of variational principles in vision problems within a rigorous theoretical framework.

We wish to show that these variational principles follow in a natural and rigorous way from the ill-posed nature of early vision problems. We will then propose a general framework for "solving" many of the processes of early vision.

Ill-Posed Problems

In 1923, Hadamard defined a mathematical problem to be *well-posed* when its solution

(a) exists

(b) is unique

(c) depends continuously on the initial data (this means that the solution is robust against noise).

Most of the problems of classical physics are well-posed, and Hadamard argued that physical problems had to be well-posed. "Inverse" problems, however, are usually ill-posed. Inverse problems can usually be obtained from the direct problem by exchanging the role of solution and data. Consider, for instance,

$$y = Az \quad (1)$$

where A is a known operator. The direct problem is to determine y from z , the inverse problem is to obtain z when y ("the data") are given. Though the direct problem is usually well-posed, the inverse problem is usually ill-posed, when z and y belong to a Hilbert space¹.

Typical ill-posed problems are analytic continuation, back-solving the heat equation, superresolution, computer tomography, image restoration and the determination of the shape of a drum from its frequency of vibration, a problem which was made famous by Marc Kac (1966). In early vision, most problems are ill-posed because the solution is not unique (but see later the case of edge detection).²

Regularization Methods

Rigorous regularization theories for "solving" ill-posed problems have been developed during the past years (see especially Tikhonov, 1963; Tikhonov and Arsenin, 1977; and Nashed, 1974, 1976). The basic idea of regularization techniques is to restrict the space of acceptable solutions by choosing the function that minimizes an appropriate functional. The regularization of the ill-posed problem of finding z from the data y such that $Az = y$ requires the choice of norms $\| \cdot \|$ (usually quadratic) and of a *stabilizing functional* $\|I'z\|$. The choice is dictated by mathematical considerations, and, most importantly, by a physical analysis of the generic constraints on the problem. Three main methods can then be applied (see Bertero, 1982):

I) Among z that satisfy $\|Pz\| \leq C$ —where C is a constant—, find z that minimizes

$$\|Az - y\|, \quad (2)$$

II) Among z that satisfy $\|Az - y\| \leq C$, find z that minimizes

$$\|Pz\|, \quad (3)$$

III) Find z that minimizes

$$\|Az - y\|^2 + \lambda \|Pz\|^2, \quad (4)$$

where λ is a regularization parameter.

The first method consists of finding the function z that satisfies the constraint $\|Pz\| \leq C$ and best approximates the data. The second method computes the function z that is sufficiently close to the data (C depends on the estimated errors and is zero if the data are noiseless) and is most "regular". In the third method, the regularization parameter λ controls the compromise between the degree of regularization of the solution and its closeness to the data. Regularization theory provides techniques to determine the best λ (Tikhonov and Arsenin, 1977; Wahba, 1980). It also provides a large body of results about the form of the stabilizing functional P that ensure uniqueness of the result and convergence. For instance, it is possible to ensure uniqueness in the case of Tikhonov's stabilizing functionals (also called stabilizers of p -th order) defined by

$$\|Pz\|^2 = \int_{\xi=0}^p c_r(\xi) \left(\frac{d^r z}{d\xi^r} \right)^2 d\xi, \quad (5)$$

where $c_r(\xi)$ are positive weighting factors. Equation (5) can be extended in the natural way to several dimensions. If one seeks regularized solutions of eq.(1) with P given by eq. (5) in the Sobolev space W_2^p of functions that have square-integrable derivatives up to p -th order, the solution can be shown to be unique (up to the null space of P), if A is linear and continuous. This is because for every p the space W_2^p is a Hilbert space and $\|Pz\|^2$ is a quadratic functional (see theorem 1, Tikhonov and Arsenin, 1977; p. 63). It turns out that most stabilizing functionals used so far in early vision are of the Tikhonov type

(see also Terzopoulos, 1984a,b).³ They all correspond to either interpolating or approximating splines (for method II and method III, respectively).

Example I: Motion

Our first claim is that variational principles introduced recently in early vision for the problem of shape from shading, computation of motion, and surface interpolation are exactly equivalent to regularization techniques of the type we described. The associated uniqueness results are directly provided by regularization theory. We briefly discuss the case of motion computation in its recent formulation by Hildreth (1984a,b).

Consider the problem of determining the two-dimensional velocity field along a contour in the image. Local motion measurements along contours provide only the component of velocity in the direction perpendicular to the contour. Figure 1 shows how the local velocity vector $V(s)$ is decomposed into a perpendicular and a tangential component to the curve

$$V(s) = v^T(s)T(s) + v^\perp(s)N(s) \quad (6)$$

The component v^\perp and direction vectors $T(s)$ and $N(s)$, are given directly by the initial measurements, the "data". The component $v^T(s)$ is not and must be recovered to compute the full two-dimensional velocity field $V(s)$. Thus the "inverse" problem of recovering $V(s)$ from the data v^\perp is ill-posed because

the solution is not unique. Mathematically, this arises because the operator K defined by

$$v^\perp = KV \quad (7)$$

is not injective. Equation (7) describes the imaging process as applied to the physical velocity field V which consists of the x and y components of the velocity field on the image plane.

Intuitively, the set of measurements given by $v^\perp(\cdot)$ over an extended contour should provide considerable constraint on the motion of the contour. An additional generic constraint, however, is needed to determine this motion uniquely. For instance, rigid motion on the plane is sufficient to determine V uniquely but is very restrictive, since it does not cover the case of motion of a rigid object in space. Hildreth suggested, following Horn and Schunck (1981), that a more general constraint is to find the smoothest velocity field among the set of possible velocity fields consistent with the measurements. The choice of the specific form of this constraint was guided by physical considerations — the real world consists of solid objects with smooth surfaces whose projected velocity field is usually smooth — and by mathematical considerations — especially uniqueness of the solution. Hildreth proposed two algorithms: in the case of exact data the functional to be minimized is a measure of the smoothness of the velocity field

$$\|PV\|^2 = \int \left(\frac{\partial V}{\partial s} \right)^2 ds \quad (8)$$

subject to the measurements $v^\perp(s)$. Since in general there will be error in the measurements of v^\perp , the alternative method is to find V that minimizes

$$\|KV - v^\perp\|^2 + \lambda \int \left(\frac{\partial V}{\partial s} \right)^2 ds. \quad (9)$$

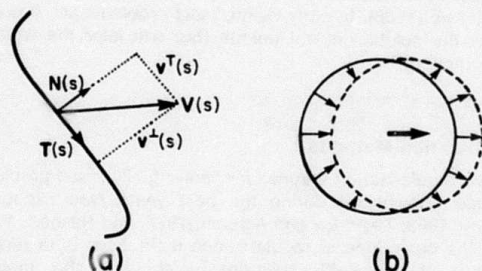


Figure 1. Decomposition and ambiguity of the velocity field. a) The local velocity vector $V(s)$ in the image plane is decomposed according to eq.(6) into components perpendicular and tangent to the curve. b) Local measurements cannot measure the full velocity field: the circle undergoes pure translation: the arrows represent the perpendicular components of velocity that can be measured from the images. From Hildreth, 1984a.

It is immediately seen that these schemes correspond to the second and third regularizing method respectively. Uniqueness of the solutions (proved by Hildreth¹ for the case of equation (8)) is a direct consequence for both equations (8) and (9) of standard theorems of regularization theories. In addition, other results can be used to characterize how the correct solution converges depending on the smoothing parameter λ .

Example II: Edge Detection

We have recently applied regularization techniques to another classical problem of early vision – edge detection. Edge detection, intended as the process that attempts to detect and localize changes of intensity in the image (this definition does not encompass all the meanings of edge detection) is a problem of numerical differentiation (Torre and Poggio, 1984). Notice that

differentiation is a common operation in early vision and is not restricted to edge detection. The problem is ill-posed because the solution does not depend continuously on the data.⁵ The intuitive reason for the ill-posed nature of the problem can be seen by considering a function $f(x)$ perturbed by a very small (in L_2 norm) "noise" term $\epsilon \sin \Omega x$. $f(x)$ and $f(x) + \epsilon \sin \Omega x$ can be arbitrarily close, but their derivatives may be very different if Ω is large enough.

In 1-D, numerical differentiation can be regularized in the following way. The "image" model is $y_i = f(x_i) + \epsilon_i$, where y_i is the data and ϵ_i represent errors in the measurements. We want to estimate f' . We chose a regularizing functional $\|f'\| = \int (f''(x))^2 dx$, where f'' is the second derivative of f . The second regularizing method (no noise in the data) is equivalent then to using interpolating cubic splines for differentiation.⁶ The third regularizing method, which is more natural since it takes into account errors in the measurements, leads to the variational problem of minimizing

$$\sum (y_i - f(x_i))^2 + \lambda \int (f''(x))^2 dx. \quad (10)$$

Poggio et al. (1984) have shown (a) that the solution of this problem f can be obtained by convolving the data y_i (assumed on a regular grid) with a convolution filter R , and (b) that the filter R is a cubic spline⁷ with a shape very close to a Gaussian and a size controlled by the regularization parameter λ (see figure 2). Differentiation can then be accomplished by convolution of the data with the appropriate derivative of this filter. The optimal value of λ can be determined for instance by cross validation and other techniques. This corresponds to finding the optimal scale of the filter.⁸

These results can be directly extended to two dimensions to cover both edge detection and surface interpolation and approximation. The resulting filters are very similar to two of the edge detection filters derived and extensively used in recent years (Marr and Hildreth 1980; Canny, 1983; see Torre and Poggio, 1984).

Other problems in early vision such as shape from shading (Ikeuchi and Horn, 1981) and surface interpolation (Grimson, 1981b 1982; Terzopoulos, 1983, 1984), in addition to the computation of velocity, have already been formulated and "solved" in similar ways using variational principles of the type suggested by regularization techniques (although this was not realized at the time). It is also clear that other problems such as stereo⁹ and structure from motion¹⁰ can be approached in terms of regularization analysis.

Physical Plausibility of the Solution

Uniqueness of the solution of the regularized problem—which is ensured by formulations such as equations (2)-(4) – is not the only (or even the most relevant) concern of regularization analysis.

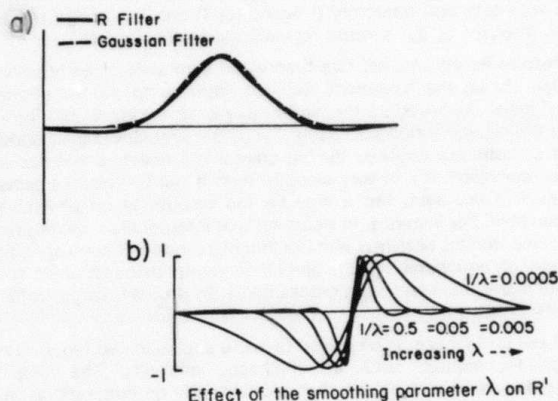


Figure 2. The edge detection filter. a) The convolution filter obtained by regularizing the ill-posed problem of edge detection with method (III) (see Poggio et al., 1984). It is a cubic spline (solid line), very similar to a gaussian (dotted line). b) The first derivative of the filter for different values of the regularizing parameter λ , which effectively controls the scale of the filter. This one-dimensional profile can be used for two-dimensional edge detection by filtering the image with oriented filters with this transversal cross-section and choosing the orientation with maximum response (see Canny, 1983).

Physical plausibility of the solution is the most important criterion. The decision regarding the choice of the appropriate stabilizing functional cannot be made judiciously from purely mathematical considerations. A physical analysis of the problem and of its generic constraints have the upper hand. Regularization theory provides a framework within which one has to seek constraints that are rooted in the physics of the visual world. This is, of course, the challenge of regularization analysis. Conditions characterizing the physically correct solutions can be derived¹¹ (for the case of motion, see Yuille, 1983 and for edge detection, see Poggio et al., 1984).

From a more biological point of view, a careful comparison of the various "regularization" solutions with human perception promises to be a very interesting area of research, as suggested by Hildreth's work. For some classes of motions and contours, the solution of equations (8) and (9) is not the physically correct velocity field. In these cases, however, the human visual system also appears to derive a similar, incorrect velocity field (Hildreth, 1984a,b).

Conclusion

The concept of ill-posed problems and the associated regularization theories seem to provide a satisfactory theoretical framework for part of early vision. This new perspective justifies the use of variational principles of a certain type for solving specific problems, and suggests how to approach other early vision problems.

It provides a link between the computational (ill-posed) nature of the problems and the computational structure of the solution (as a variational principle). In a companion paper (Poggio and Koch, 1984), we will discuss computational "hardware" that is natural for solving variational problems of the type implied by regularization methods. The approach can be extended to other sensory modalities and to some motor control problems. For instance, a recently proposed solution to the problem of executing a voluntary arm trajectory (Hogan, 1984) can be recognized as an instance of our second regularization technique.¹²

Despite its attractions, this theoretical synthesis of early vision also shows the limitations that are intrinsic to the variational solutions proposed so far, and in any case to the simple forms of the regularization approach. The basic problem is the degree of smoothness required for the unknown function z that has to be recovered. If z is very smooth, then it will be robust against noise in the data, but it may be too smooth to be physically plausible. For instance, in visual surface interpolation, the degree of smoothness obtained with the thin plate model (from a specific form of equations (4)-(5)) smoothes depth discontinuities too much and often leads to unrealistic results (but see Terzopoulos, 1984).

These problems may be solved by more sophisticated regularization techniques, such as stochastic methods. The simple regularization techniques analyzed here rely on quadratic variational principles that lead to linear Euler-Lagrange equations. Thus the solution can be found by filtering the data through an appropriate linear filter. Analog electrical or chemical networks can be devised for the specific variational principles (Poggio and Koch, 1984). Again, the universe of solutions to quadratic variational principles is somewhat restricted.

Nonquadratic variational principles are, however, possible. They may arise naturally in one of the most fundamental problems in early vision, the problem of integrating different sources of information, such as stereo, motion, shape from shading, etc. This problem is ill-posed, not just because the solution is not unique (the standard case), but because the solution is usually overconstrained and may not exist. The use and extensions of tools from regularization theory to analyze the fusion of information from different sources is one of the most interesting challenges in the theory of early vision.^{13,14}

The problem is related to the deep question of the computational organization of a visual processor and its control structure. It is unlikely that variational principles alone could have enough flexibility to control and coordinate the different modules of early vision and their interaction with higher level knowledge. This also hints at the basic limitation of regularization methods that makes them suitable only for the first stages of vision. They derive numerical representations—surfaces—from numeri-

cal representations—images. It is difficult to see how the computation of the more symbolic type of representations that are essential for a powerful vision processor can fit into this theoretical framework¹⁵.

In summary, we have outlined a new theoretical framework that from the computational nature of early vision leads to algorithms for solving them, and suggests a specific class of appropriate hardware. The common computational structure of many early vision problems is that they are ill-posed in the sense of Hadamard. Regularization analysis can be used to solve them in terms of variational principles of a certain type that enforce constraints derived from a physical analysis of the problem. Analog networks—whether electrical or chemical—are a simple and attractive way of solving this type of variational principles.

Acknowledgments: The idea of vision problems as ill-posed problems originated from a conversation with Professor Mario Bertero of the University of Genoa about the edge-detection work with V. Torre. Work on visual problems as ill-posed problems is now going on at MIT with Alan Yuille, Christof Koch, Harry Voorhees and at the University of Genoa with Alessandro Verri. The idea of formulating vision problems in terms of variational principles is due to a number of people, mostly at the A.I. Lab at MIT. Among them, we would like to mention Eric Grimson, Demetri Terzopoulos, Berthold Horn, Shimon Ullman, Mike Brady, Ellen Hildreth and Alan Yuille. We are grateful to A. Yuille, E. Hildreth, D. Terzopoulos and C. Koch for many discussions and comments and for critically reading the manuscript. We are afraid to forget Carol Bonomo. This report describes research done within the Artificial Intelligence Laboratory and the Center for Biological Information Processing (Whitaker College) at the Massachusetts Institute of Technology. Support for the A. I. Laboratory's research in artificial intelligence is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505. The Center's support is provided in part by the Sloan Foundation and in part by Whitaker College. Funds for collaborative travel were provided by NATO grant 237.81. Dr. Torre is at the Institute of Physics, University of Genoa, Italy.

Footnotes

[1] Whether a problem is well- or ill-posed depends on the triplet (A, Z, Y) where Z and Y are the solution and the data space respectively.

[2] The reason for the lack of uniqueness is that the operator corresponding to A is usually not injective, as in the case of shape from shading, surface interpolation and computation of motion.

To clarify some of the structure of ill-posed problems, let us consider the linear operator

$$Az = y. \quad (1)$$

If z and y are finite vectors, then the inverse problem is easily solved by finding the inverse of A , or its pseudoinverse. It is well known that if A is a square matrix, A^{-1} exists if $\det|A| \neq 0$.

Now let us suppose that $z \in Z$ and $y \in Y$, where Z and Y are Hilbert spaces. The inverse problem is well-posed iff the three conditions of Hadamard are satisfied. In particular,

(1) condition (a) of Hadamard is satisfied iff the range of A is $R(A) = Y$.

(2) condition (b) of Hadamard is satisfied iff A is injective.

(3) condition (c) of Hadamard is satisfied iff $R(A)$ is a closed set.

If the operator A is compact and $R(A)$ does not have finite dimensions, $R(A)$ is open, and therefore the inverse problem is also ill-posed.

Most linear operators whose domain and co-domain are Hilbert spaces are compact operators. In fact, if E and F are measurable, bounded sets $E \in \mathcal{R}^n$ and $F \in \mathcal{R}^m$, and $k(t, s)$ is a measurable function defined on $E \times F$, then the linear operator $A: L_2(E) \rightarrow L_2(F)$ defined as

$$(Az)(t) = \int k(t, s)z(s)ds \quad (2)$$

is compact and $R(A)$ has finite dimensions iff $k(t, s)$ is separable, i.e.,

$$k(t, s) = \sum_{k=1}^n \alpha_k(t) \beta_k(s). \quad (3)$$

Obviously if $R(A)$ has finite dimension, then $R(A)$ cannot coincide with Y , and therefore the inverse problem of an integral operator or a convolution is in general an ill-posed problem.

We can relax condition (2) and admit the case that A is not injective. The problem is then regularized by introducing an appropriate norm and finding the generalized pseudoinverse of the inverse problem (1).

When y is not in $R(A)$, it is not easy to regularize the problem without altering the essence of the problem itself.

[3] J. Canny's (1983) variational formulation can be derived from eq. (4) and a stabilizing functional of the form of eq. (5) (see Poggio et al., 1984).

[4] It is shown in Hildreth (1984a) that extremizing equation (8) yields a unique velocity field, since it corresponds to minimizing a positive definite functional on a convex set. The theorems of du Bois-Reymond state that, provided $\frac{\partial V}{\partial x}$ is continuous the solution of the minimization problem will be the solution of the corresponding Euler-Lagrange equations.

[5] The problem is to find the solution z to

$$y = Az$$

with $(Az)(x) = \int_0^x z(s) ds$. Thus, z is the derivative of the data y . The problem is (mildly) ill-posed because if $z \in L_2[0, 1]$, the compact operator A is not closed in $L_2[0, 1]$.

[6] For data on a regular grid, it corresponds to convolving the data with the L_4 filters of Schoenberg (1946).

[7] A higher degree stabilizer may be used for higher derivatives, leading to higher order splines.

[8] Methods such as the Generalized Cross Validation method (GCV) (Wahba, 1980; see also Reinsch, 1967) may be used to find the "optimal" scale of the filter, i.e., the optimal λ . Fingerprints (Yuille and Poggio, 1983) may provide a method for finding the optimal value of the regularization parameter λ . This follows from the fact that the filter given by equation (10) is very similar to a Gaussian and that λ effectively controls the scale of the filter (see Poggio et al., 1984).

[9] Another clearly ill-posed problem is stereo-matching. It is not immediately obvious, however, what the correct regularizing procedure is. Berthold Horn has suggested (personal communication) a variational principle for stereo-matching similar to his scheme for computing optical flow. The norm to be minimized measures deviations from smoothness of the disparity field. Specifically, the norm of the derivative of the z component, the depth component, has to be minimized subject to the constraints given by the data. This can be regarded again as a variational principle of the type that is obtained directly using the standard regularization methods of ill-posed problems. We are presently developing regularization solutions to the stereo problem (Yuille and Poggio, in preparation).

The problem of shape-from-contours in the variational formulation of Brady and Yuille (1983) is an ill-posed problem but the solution is not of the standard regularization type.

[10] The rubbery constraint proposed by Ullman (1983) is more general than the rigidity constraint. It may be possible to reformulate it according to regularization techniques.

[11] A method for checking physical plausibility of a variational principle is, of course, computer simulation. A simple technique

we suggest is to use the Euler-Lagrange equation associated with the variational problem.

In the computation of motion, Yuille (1983) has obtained the following *sufficient and necessary* condition for the solution of the variational principle equation (8), to be the correct physical solution

$$T \cdot \frac{\partial^2 V}{\partial s^2} = 0$$

where T is the tangent vector to the contour and V is the true velocity field. The equation is satisfied by uniform translation or expansion and by rotation only if the contour is polygonal. These results suggest that algorithms based on the smoothness principle will give correct results, and hence be useful for computer vision systems, when (a) motion can be approximated locally by pure translation, rotation or expansion, or (b) objects have images consisting of connected straight lines. In other situations, the smoothness principle will not yield the correct velocity field, but may yield one that is qualitatively similar and close to human perception (Hildreth, 1984a,b).

In the case of edge detection (intended as numerical differentiation), the solution is correct if and only if the intensity profile is a polynomial spline of odd degree greater than three (Poggio et al., 1984).

[12] The variational principle (minimization of jerk) corresponds to the second regularization method, with $P = d^3/dx^3$. The associated interpolating function is a quintic spline. Analog networks for solving the problem can be devised (Poggio and Koch, 1984). It may be interesting to consider our third method of regularization in the context of the available data on arm trajectories.

[13] The variational principles that we have considered so far for early vision processes are quadratic and lead therefore to linear equations. The ill-posed problem of combining several different sources of surface information may easily lead to non-quadratic regularization expressions (though different "non-interacting" constraints can be combined in a convex way, see Terzopoulos, 1984). These minimization problems will in general have multiple local minima. Schemes similar to annealing (Kirkpatrick, Gelatt and Vecchi 1983; Hinton and Sejnowski, 1983; Geman and Geman, 1984) may be used to find the global minimum (see also Poggio and Koch, 1984).

[14] This is a list of open problems on which we are presently working:

- a) Regularized solution for stereo matching.
- b) Regularized solution for structure from motion.
- c) Full extension of the edge detection analysis to 2-D and application to surface approximation for computing differential properties of surfaces.
- d) Analysis and implementation of methods for finding the optimal regularization parameter λ . Use of fingerprints.
- e) Connection between the regularizing parameter λ , the iteration number in iterative regularizing methods (Nashed, 1976) and the truncation of a formal power series expansion of the regularizing operator.
- f) Use of stochastic regularization methods (see also Geman and Geman, 1984).

[15] But see Hummel and Zucker, 1980.

References and related papers

- Baker, H. H. and Binford, T. O., "Depth from Edge and Intensity Based Stereo," *Seventh International Joint Conference on Artificial Intelligence*, August 1981a, 631-636.
- Barrow, H.G. and Tenenbaum, J.M. "Interpreting line drawings as three dimensional surfaces" *Artif. Intelligence*, 17, 75-117, 1981.
- Bertero, M. "Problemi lineari non ben posti e metodi di regolarizzazione", in: *Problem non ben posti ed inversi*, Istituto di Analisi Globale, Firenze, 1982.
- Brady, J.M., Grimson, W.E.L., and Langridge, D.J. "Shape encoding and subjective contours" *First Annual Natl. Conf. on Artificial Intelligence* 15-17, 1980
- Brady, J.M., and Yuille, A., "An extremum principle for shape from contour", *I.E.E.E Trans. on Pattern Analysis and Machine Intelligence*, 288,301, 1984.
- Canny, J. F., "Finding Edges and Lines in Images," Massachusetts Institute of Technology Artificial Intelligence Laboratory Technical Report TR-720, June 1983.
- Courant, R., and Hilbert, D., *Methods of Mathematical Physics*, Interscience Publishers, London, 1962.
- Geman, S. and Geman, D. "Stochastic Relaxation, Gibbs Distributions and the Bayesian Restoration of Images", *PAMI*, in press, 1984.
- Grimson, W.E.L., "A computer implementation of a theory of human stereo vision", *Phil. Trans. R. Soc. Lond., B*, 292, 217-253, 1981a.
- Grimson, W.E.L., *From Images to Surfaces: A computational study of the human early visual system*, MIT Press, Cambridge, Ma., 1981b.
- Grimson, W.E.L., "A computational theory of visual surface interpolation", *Phil. Trans. R. Soc. Lond., B*, 298, 395-427, 1982.
- Hadamard, J., *Lectures on the Cauchy Problem in Linear Partial Differential Equations*, Yale University Press, New Haven, 1923.
- Hildreth, E.C., "Implementation of a theory of edge detection," S.M. Thesis, Department of Computer Science and Electrical Engineering, Massachusetts Institute of Technology, 1980. (see also MIT AI Lab Technical Report 597, 1980).
- Hildreth, E.C., *The Measurement of Visual Motion*, MIT Press, Cambridge, Massachusetts, 1984a.
- Hildreth, E.C., "Computation of the velocity field", *Proc. R. Soc. Lond. B*, 221, 189-220, 1984b.
- Hinton, G.E., and Sejnowski, T.J., "Optimal perceptual inference", *Proc. IEEE 1983 Conf. Computer Vision and Pattern Recognition*, Washington, DC, 1983.
- Hogan, N. "An organising principle for voluntary movements". Submitted to *J. Neurosci.*, 1984.
- Horn, B.K.P., "Determining lightness from an image", *Computer Graphics and Image Processing*, 3, 111-299, 1974.
- Horn, B.K.P., "The least energy curve", *AI Memo 612*, MIT AI Lab., Cambridge, MA, 1981
- Horn, B.K.P., and Schunck, B.G., "Determining optical flow", *Artificial Intelligence*, 17, 185-203, 1981.
- Hummel, R. and Zucker, S. "On the foundations of relaxation labeling processes." McGill University, Computer Vision and Graphics Laboratory, TR 80-7, 1980.
- Ikeuchi, K., and Horn, B.K.P., "Numerical shape from shading and occluding boundaries", *Artificial Intelligence*, 17, 141-184, 1981.
- Kac, M. "Can one hear the shape of a drum?" *Am. Math. Monthly*, 73, No. 4, Part II, 1-23, 1966.
- Kass, M., "A computational framework for the visual correspondence problem", *Proc. 8th Int. J. Conf. AI*, Karlsruhe, W. Germany, 1043-1045, 1983.
- Kass, M.H. "Computing stereo correspondance", *M. S., Mass. Institute of Technology*, 1984.
- Kirkpatrick, S., Gelatt, C.D., Jr., Vecchi, M.P., "Optimization by simulated annealing", *Science*, 220, 671-680, 1983.
- Marr, D., and Hildreth, E.C., "Theory of edge detection", *Proc. R. Soc. Lond. B*, 207, 187-217, 1980.
- Marr, D. and Poggio, T. "Cooperative computation of stereo disparity", *Science*, 194, 283-287, 1976.
- Marr, D. and Poggio, T. "A theory of human stereo vision," *Proc. Roy. Soc. Lond. B* 204 (1979), 301-328. (an earlier version appeared as MIT AI Lab Memo 451, 1977).
- Mayhew, J.E.W. and Frisby, J.P. "Psychophysical and computational studies towards a theory of human stereopsis," *Artificial Intelligence*, 17, 349-385, 1981.
- Nashed, M. Z., "Approximate regularized solutions to improperly posed linear integral and operator equations in *Constructive and Computational Methods for Differential and Integral Equations* G. Auger, ed., Akademie Verlag, Berlin, 289-296, 1974.
- Nashed, M. Z., ed. *Generalized Inverses and Applications*, Academic Press, New York, 1976.
- Nishihara, H.K. and Poggio, T., "Stereo vision for robotics", *Proceedings of the First International Symposium of Robotics Research*, in press.
- Poggio, T. and Koch, C., "Analog networks: a new approach to neuronal computation", *Artificial Intelligence Lab. Memo, No. 783*. MIT, Cambridge, MA, 1984.
- Poggio, G. and Poggio, T., *Annual Review of Neuroscience*, 7, 379-412, 1984.
- Poggio, T., Voorhees, H. and Yuille A., "Regularizing Edge Detection", *Artificial Intelligence Lab. Memo, No. 776*, MIT, Cambridge, 1984.
- Reinsch, C.H., "Smoothing by spline functions", *Numer. Math.*, 10, 177-183, 1967.
- Schoenberg, I. J., "Contributions to the problem of approximation of equidistant data by analytic functions." *Quart. Appl. Math.*, 4, 45-99, 112-141, 1946.
- Schunck, B.G., and Horn, B.K.P., "Constraints on optical flow computation", *Proc. IEEE Conf. Pattern Recognition and Image Processing*, 205-210, 1981.
- Terzopoulos, D., "Multilevel computational processes for visual surface reconstruction", *Computer Vision, Graphics, and Image Processing*, 24, 52-96, 1983.
- Terzopoulos, D., "Multiresolution computation of visible-surface representations," Ph.D. Thesis, Dept of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, January, 1984a.
- Terzopoulos, D., "Multilevel reconstruction of visual surfaces: variational principles and finite element representations" *Artificial Intelligence Lab. Memo 671*, reprinted in *Multiresolution Image Processing and Analysis*, A. Rosenfeld (ed.), 237-310, Springer-Verlag, 1984b.

Terzopoulos, D., "Integrating visual information from multiple sources for the cooperative computation of surface shape", to appear in *From pixels to predicates: recent advances in computational and robotic vision*, ed. A. Pentland, Ablex, 1985.

Tikhonov, A. N. "Solution of incorrectly formulated problems and the regularization method", *Soviet Math. Dokl.*, **4**, 1035-1038, 1963.

Tikhonov, A.N. and Arsenin, V.Y., *Solutions of ill-posed problems*, Winston & Sons, Washington, D.C., 1977.

Torre, V. and Poggio, T., "On Edge Detection", *Artificial Intelligence Lab. Memo*, No. 768, MIT, Cambridge, MA, March 1984.

Ullman, S., "Filling in the gaps: The shape of subjective contours and a model for their generation" *Biological Cybernetics* **25**, 1-6, 1976.

Ullman, S., "The interpretation of structure from motion", *Proc. R. Soc. Lond. B*, **203**, 405-426, 1979.

Ullman, S., "Maximizing rigidity: the incremental recovery of 3-D structure from rigid and rubbery motion" *Artificial Intelligence Lab. Memo*, No. 721, MIT, Cambridge, MA, June 1983.

Wahba, G., "Ill-posed problems: Numerical and statistical methods for mildly, moderately and severely ill-posed problems with noisy data", *Tech. Rep.*, No. 595, Univ. of Wisconsin, Madison, 1980.

Yuille, A., "The smoothest velocity field and token matching schemes" *Artificial Intelligence Lab. Memo*, No. 724, MIT, Cambridge, MA, August 1983.

Yuille, A., "Zero crossings on lines of curvature", *Artificial Intelligence Lab. Memo*, No. 718, MIT, Cambridge, MA, 1984.

Yuille, A.L. and Poggio, T. "Fingerprints Theorems for Zero-crossings", *Artificial Intelligence Lab. Memo*, No. 730, MIT, Cambridge, MA, 1983b.

Yuille, A.L. and Poggio, T. "A generalized Ordering Constraint for Stereo Correspondance", *Artificial Intelligence Lab. Memo*, No. 777, MIT, Cambridge, MA, 1984.

Recovering the Camera Parameters from a Transformation Matrix

Thomas M. Strat
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025

Abstract

The transformation of the three-dimensional coordinates of a point to the two-dimensional coordinates of its image can be expressed compactly as a 4×4 homogeneous coordinate transformation matrix in accordance with a particular imaging geometry. The matrix can either be derived analytically from knowledge about the camera and the geometry of image formation, or it can be computed empirically from the coordinates of a small number of three-dimensional points and their corresponding image points. Despite the utility of the matrix in image understanding, motion tracking, and autonomous navigation, very little is understood about the inverse problem of recovering the projection parameters from its coefficients. Previous attempts have produced solutions that require iteration or the solution of a set of simultaneous nonlinear equations. This paper shows how the location and orientation of the camera, as well as the other parameters of the image-formation process can easily be computed from the homogeneous coordinate transformation matrix. The problem is formulated as a simple exercise in constructive geometry and the solution is both noniterative and intuitively understandable.

1 Introduction

Homogeneous coordinates and the *homogeneous coordinate transformation matrix* are a convenient means for representing arbitrary transformations, including perspective projection in a single formalism. One such use for this matrix is as a *camera transformation matrix* that maps points in an object-centered coordinate system into the corresponding points in image coordinates according to a particular imaging geometry [7]. The camera transformation matrix has seen wide use in several disciplines. Rogers and Adams present numerous applications in computer graphics [8]. Other fields that have made use of the camera transformation matrix include stereo reconstruction, robot vision, photogrammetry, unmanned-vehicle guidance, and image understanding [5], [6], [12], [11]. Several techniques for computing the matrix have been derived, yet very little is understood about how to recover the projection parameters from the coefficients of the matrix.

When the location and orientation of the camera are known, the camera transformation matrix that models the image formation process can easily be derived analytically [1]. This model forms the basis for subsequent processing of images produced by that camera. On the other hand, when the location and orientation of the camera are unknown, the parameters of image formation must be derived from the correspondence between a set of image features and a set of object features. Images obtained from an unknown source or from cameras mounted on moving

platforms exemplify contexts in which the imaging geometry may not be known.

One approach to computing the parameters of the image formation process directly is embodied in RANSAC, developed by Fischler and Bolles [3]. RANSAC computes the camera location directly from a set of landmarks with known three-dimensional locations when, in addition, the focal length and piercing point are known.

Alternatively, several methods exist for estimating the coefficients of the camera transformation matrix from the correspondence between image and object coordinates. Sutherland [10] describes a method to determine the matrix experimentally from the image by using a least-squares technique to obtain the coefficients from available ground truth data. A consideration of the experimental errors involved and a means for improving accuracy are described by Sobel [9].

The issue addressed in this paper is how to determine the imaging geometry from a camera transformation matrix that has been derived experimentally. For example, given a photograph taken by an unknown camera from an unknown location and which, moreover, may have been cropped and/or enlarged, how can we recover the camera's position and orientation and determine the extent to which the picture was cropped or enlarged? If some ground truth data are available, an established least-squares technique such as Sutherland's can be used to derive the camera transformation matrix, whereupon the problem reduces to that of computing the values of the desired parameters from the matrix. Ganapathy [4] recently published the first noniterative method for solving this problem by posing it as a set of eleven simultaneous nonlinear equations that can be solved to obtain the eleven independent coefficients of the camera transformation matrix. While the method is successful at solving for camera location and orientation, it is an algebraic one that provides little insight into the underlying geometry. The method to be described here is a geometric one that solves for the same parameters, but is posed as a simple problem in constructive geometry and allows an intuitively clear derivation.

This work has immediate application in several areas:

- Many algorithms in image understanding require knowledge of the camera parameters. These can be computed from an arbitrary photograph by using the method presented here when ground truth data is available.

- Autonomous navigation can be posed as a problem in deriving camera parameters. A cruise missile, for instance, could obtain the camera transformation matrix from a terrain model stored on board and then compute the camera parameters that define the vehicle's location and heading.
- A stationary camera viewing a robot arm workspace could determine the position and orientation of the arm. Conspicuous marking of several points on a part of the manipulator would allow their easy extraction from an image and provide the ground truth necessary for Sutherland's algorithm to ascertain the camera transformation matrix. The camera parameters can be derived from this matrix, and the location and orientation of the manipulator can then be obtained relative to the stationary camera.

2 The Camera Transformation Matrix

As indicated earlier, the camera transformation matrix can be used to model in a single formalism the effects of rotation, translation, perspective, scaling, and cropping—i.e., all the variables associated with the normal imaging process. Here we review the fundamentals of homogeneous coordinate transformations that are essential for understanding the decomposition to be described. The presence of an ideal lens and the absence of any atmospheric distortions are assumed.

The imaging situation can be modeled as shown in Figure 1. The XYZ coordinate system represents the world or object-centered coordinates. The center of projection (the location of the lens) is shown as a point L in space. The image plane is a plane between the lens and the object onto which the object is projected to obtain the image. Each image point is that point in the image plane where the plane intersects the line connecting L with the corresponding object point. The UVW coordinate system is situated such that (u, v) are the image coordinates of an image point and $w = 0$ defines the image plane. The perpendicular distance between L and the image plane is the camera focal length, f .

In a homogeneous coordinate system, a three-dimensional point (x, y, z) is represented as a four-component row vector, (tx, ty, tz, t) ; the three-dimensional coordinates are obtained by dividing through by the fourth component. A point in the world is represented as a four-component row vector and its projection in the image is obtained by postmultiplying by the 4×4 camera transformation matrix:

$$\begin{aligned} xM &= u \\ (x, y, z, 1)M &= (su, sv, sw, s) \end{aligned}$$

This homogeneous coordinate system is most useful for modeling the effects of perspective projection—further details can be found in Ballard and Brown [1].

The matrix M can be viewed as being composed of several simple transformations. While it is possible to decompose the matrix in a variety of ways, the particular decomposition chosen must capture all the degrees of freedom of the imaging geometry. The somewhat arbitrary choice used throughout this paper is shown below:

$$M = (\text{translate})(\text{rotate})(\text{project})(\text{scale})(\text{crop})$$

$$M = TRPSC \quad (1)$$

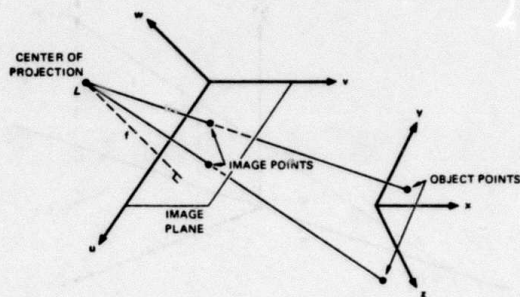


FIGURE 1 THE IMAGING GEOMETRY

Each of the component transformations can be expressed as a 4×4 matrix; multiplying them together produces the camera transformation matrix M . Details of the decomposition are given below.

2.1 Translation

Translation moves the image plane away from the object-centered origin. To translate the plane by (x_0, y_0, z_0) multiply by the matrix

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_0 & -y_0 & -z_0 & 1 \end{bmatrix}$$

2.2 Rotation

The orientation of the camera is specified by the rotation matrix R , which can be further decomposed to $R = R_x R_y R_z$, corresponding to rotation about each of the principal axes. Clockwise rotation by θ about the X axis while looking toward the origin is accomplished by

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Similarly, clockwise rotation by ϕ about the newly rotated Y axis is represented by

$$R_y = \begin{bmatrix} \cos \phi & 0 & \sin \phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \phi & 0 & \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and rotation by ψ about the new Z axis is given by

$$R_z = \begin{bmatrix} \cos \psi & -\sin \psi & 0 & 0 \\ \sin \psi & \cos \psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The first two rotations, R_x and R_y , serve to align the Z axis with the line of sight defined by the W axis. The final rotation, R_z , is within the image plane about the line of sight. Together, T and $R = R_x R_y R_z$ account for the location and orientation of the camera.

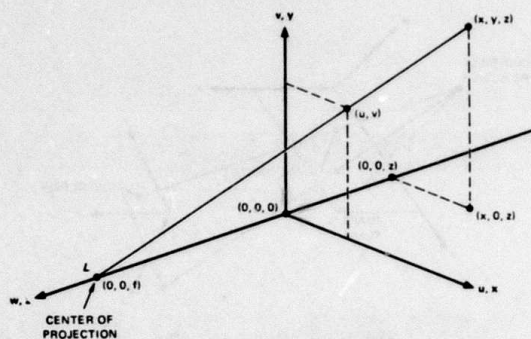


FIGURE 2 PERSPECTIVE PROJECTION AFTER ROTATION AND TRANSLATION
 $z = 0$ is the image plane.

2.3 Projection

P provides the distortion associated with perspective projection. Figure 2 shows the simplified imaging geometry after translation and rotation have been accounted for. The camera location L is on the positive Z axis at a distance f from the origin. The image plane passes through the origin and the world to be imaged lies behind the image plane. Analysis of similar triangles shows that the image coordinates

$$(u, v) = \left(\frac{fx}{f-z}, \frac{fy}{f-z} \right).$$

Using homogeneous coordinates, this perspective projection can be obtained by multiplying the homogeneous coordinates of the world point by the matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1/f \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The resulting row vector is divided through by the fourth component to renormalize the homogeneous coordinates, and then projected orthographically onto the image plane $w = 0$ to yield the proper perspective projection of the world point.

2.4 Scaling

The image coordinates can be scaled to reflect an enlargement or shrinking of the image. Scaling by k_u and k_v in the U and V directions is achieved with

$$S = \begin{bmatrix} k_u & 0 & 0 & 0 \\ 0 & k_v & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Scaling the W axis is meaningless because the perspective projection always requires that $w = 0$.

2.5 Cropping

The effect of cropping a photograph is obtained by translating the UV coordinates within the image plane. The following matrix is used to shift the origin by (u_0, v_0) :

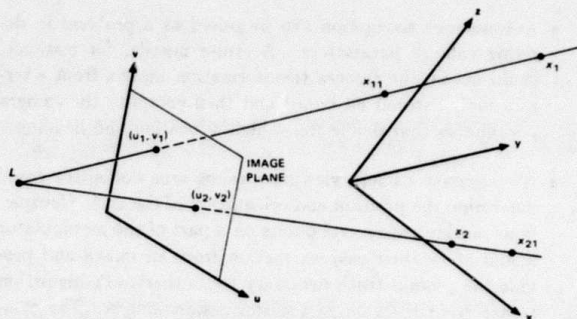


FIGURE 3 DETERMINING THE LOCATION OF THE CAMERA

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -u_0 & -v_0 & 0 & 1 \end{bmatrix}.$$

Note that neither scaling nor cropping affects the w and s coordinates of the homogeneous image point, so that orthographic projection and renormalization can take place after the entire transformation has been computed.

3 Recovering the Camera Parameters

The camera transformation matrix M allows representation of all eleven degrees of freedom associated with the image formation process. These camera parameters are embedded in the matrix in a way that makes their determination difficult. This section presents a simple method that recovers the various parameters associated with image formation. Its main advantages are that it is both noniterative and geometric, enabling a clear understanding of the equations involved.

The matrix M can be viewed as a function that maps world coordinates into image coordinates according to the constraints of Figure 1. For notational simplicity, we shall assume that all matrix multiplications automatically normalize the homogeneous coordinates of the resulting row vector. For example, $u = xM = (su, sv, sw, s) = (u, v, w, 1)$. The image formation process can then be written as

$$(u, v, 0, 1) = \text{orthoproject}(xM), \quad (2)$$

where x is the homogeneous coordinate of a world point, and $\text{orthoproject}(\cdot)$ is a function that performs an orthographic projection along the w axis such that

$$\text{orthoproject}(u, v, w, 1) = (u, v, 0, 1).$$

3.1 Location of the Camera

Figure 3 illustrates the technique for finding the center of projection. First, compute M^{-1} for later use. Note that M will always be invertible because all its components in Equation 1 are clearly invertible. The location of the center of projection, L , can be determined as follows:

Choose an arbitrary world point $x_1 = (x_1, y_1, z_1, 1)$ and compute $u_1 = x_1 M$. If we were to multiply $u_1 = (u_1, v_1, w_1, 1)$ by M^{-1} , we would obtain the original x_1 . Instead, first project u_1

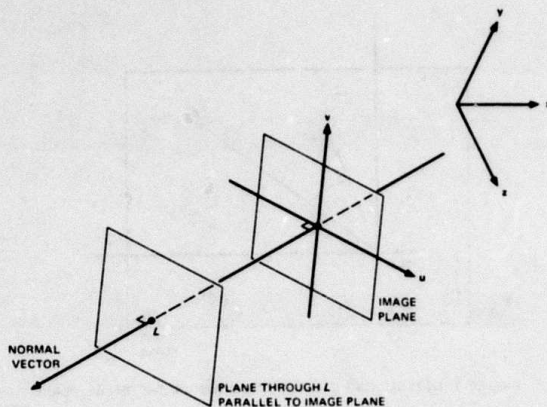


FIGURE 4 DETERMINING THE ORIENTATION OF THE CAMERA

to obtain $orthoproject(u_1) = (u_1, v_1, 0, 1)$, where (u_1, v_1) are the image coordinates of x_1 . Next, backproject this image point by multiplying by M^{-1} to obtain x_{11} . This specifies another world point, x_{11} , which is different from the original x_1 but constrained to lie along the line connecting x_1 and the center of projection L . To confirm this, note that all points lying along the line connecting x_1 and L are transformed by M to points identified by $(u_1, v_1, w, 1)$, where w varies with each point. The converse must also be true. That is, for any w , $(u_1, v_1, w, 1)M^{-1}$ specifies a point somewhere along the line connecting x_1 and L .

Repeat the above process with another point x_2 to obtain the point x_{21} , which must lie on the line connecting x_2 and L . Now x_1 and x_{11} , and x_2 and x_{21} define two lines that pass through L ; their intersection can be computed to obtain the world coordinates of L . This method will fail, of course, if either x_1 or x_2 lie in the image plane or if x_1, x_2 and L are colinear. Because their choice is arbitrary, valid points can always be found that allow the unique determination of L .

3.2 Orientation of the Camera

The orientation of the camera is defined by the orientation of the image plane (Figure 4). The latter can easily be established by observing that world points lying in the plane that is parallel to the image plane and that passes through the center of the lens will map to infinity in image coordinates. The only way this can happen for a finite world point is if the fourth component of the homogeneous image coordinate is zero.

Thus, if

$$(x, y, z, 1)M = (u, v, w, 0),$$

it follows that

$$M_{14}x + M_{24}y + M_{34}z + M_{44} = 0,$$

which is the equation of the plane through L parallel to the image plane. From this equation it is clear that the vector $n = (M_{14}, M_{24}, M_{34})$ is normal to the image plane and parallel to the camera's direction of view.

The orientation in terms of rotations about the axes can be calculated by using spherical coordinates such that

$$\theta = \arctan \frac{M_{24}}{-M_{34}}$$

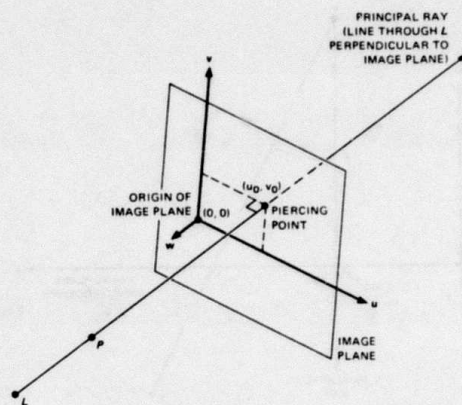


FIGURE 5 DETERMINING THE PIERCING POINT

and

$$\phi = \arcsin \frac{-M_{14}}{\sqrt{M_{14}^2 + M_{24}^2 + M_{34}^2}}$$

where θ is the clockwise rotation about the X axis and ϕ is the clockwise rotation about the rotated Y axis. The final rotation parameter, ψ , is the rotation within the image plane about the W axis. The magnitude of ψ cannot be obtained from the normal to the image plane; instead, it requires a more complex derivation that involves determination of the piercing point and the relative scale factors. These values are derived in the following sections and the value of ψ is finally computed in Section 3.5.

3.3 Piercing Point, Principal Ray, and Cropping

Much work in image understanding requires knowledge of the *piercing point* (or *stare point*) in an image. This is the point in an image that corresponds to the world point at which the camera was aimed. It is the point at which the *principal ray* (the ray along which the camera was aimed) pierces the image plane (Figure 5). The principal ray is assumed to be perpendicular to the image plane.¹

To find the piercing point, first find a point p along the principal ray (other than L):

$$p = L + k\vec{n},$$

where k is any scalar except 0. The piercing point u_0 is given by

$$u_0 = orthoproject(pM) = (u_0, v_0, 0, 1)$$

because any point along the principal ray must project to the piercing point in the image. The extent to which the image has been cropped is given by (u_0, v_0) .

3.4 Focal Length and Scale

When the center of projection is held a constant distance from the scene, there is no way to tell the difference between scaling the image and varying the focal length. For example, doubling the focal length is equivalent to enlarging the picture by a factor of two. The best one can hope for is a relation between the two

¹The image plane in some cameras used in photogrammetry is not perpendicular to the line of sight; this case, however, is not considered here.

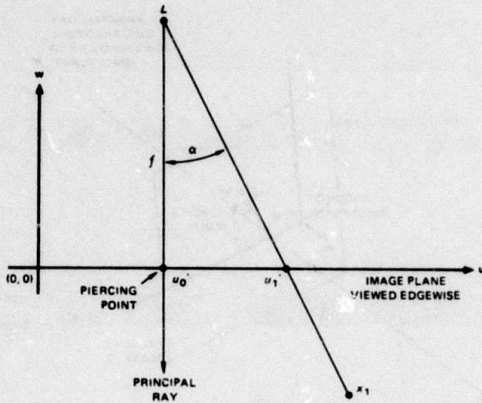


FIGURE 6 DETERMINING THE SCALE FACTOR

parameters. On the other hand, if the focal length of the camera is known, the exact scale factors can be determined.

Figure 6 shows the geometry for computing the U -component of the scale factor. First choose an arbitrary world point x_1 (not on the principal ray) and compute its image point $u_1 = \text{orthoproject}(x_1 M) = (u_1, v_1, 0, 1)$. Conversion of these image units to world units requires dividing by the scale factor such that

$$u'_1 = \frac{u_1}{k_u} \quad \text{and} \quad v'_1 = \frac{v_1}{k_v},$$

where u'_1 and v'_1 are the distances of an image point from the image origin, measured in world units. Next compute α , the angle between the principal ray and the ray from L to x_1 , projected in the plane $v=0$. Then it is clear from the diagram that the following relation must hold

$$\tan \alpha = \frac{u'_1 - u'_0}{f} = \frac{\frac{u_1}{k_u} - \frac{u_0}{k_u}}{f},$$

where k_u is the magnification of the image in the U direction. If the focal length is known, the scale factor

$$k_u = \frac{u_1 - u_0}{f \tan \alpha}$$

or if the scale factor is known,

$$f = \frac{u_1 - u_0}{k_u \tan \alpha}.$$

The computation of k_v , the V -component of the scale factor, is identical. Neither k_u nor k_v can be determined individually without knowledge of the focal length, but their ratio can be calculated from quantities derived from the matrix:

$$\frac{k_u}{k_v} = \frac{u_1 - u_0}{f \tan \alpha_u} \cdot \frac{v_1 - v_0}{f \tan \alpha_v} = \frac{(u_1 - u_0) \tan \alpha_v}{(v_1 - v_0) \tan \alpha_u}$$

3.5 Rotation within the Image Plane

We now return to the derivation of ψ , the rotation of the camera about the W axis. This rotation is equivalent to cropping the image at an angle to the UV coordinate system. The value of ψ is found by choosing a world point and comparing its transformation under two different situations, as illustrated in Figure 7.

First, use the coordinates of L computed earlier and an arbitrary focal length to reconstruct the translation matrix T . Use

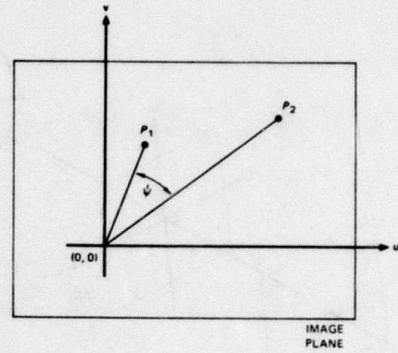


FIGURE 7 DETERMINING THE ROTATION WITHIN THE IMAGE PLANE

the values of θ and ϕ to reconstruct R_x and R_y and use the chosen focal length to construct a perspective projection matrix P' . This comprises a model $M_1 = TR_x R_y P'$, which can be employed to compute the transformation of an arbitrary world point. Call the resulting point p_1 .

Next, use the previously determined piercing point to reconstruct the matrix C . Then undo the effects of cropping from the camera model by multiplying the original camera transformation matrix by C^{-1} to obtain

$$M' = MC^{-1} = TRPSCC^{-1} = TRPS.$$

Now the effects of unbalanced scaling will be eliminated. Use the relative scale factor computed earlier to construct a scale transformation matrix:

$$S' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{k_u}{k_v} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Then multiply M' by S' to obtain

$$M_2 = M'S' = TRPSS' = TRPS'',$$

where

$$S'' = SS' = \begin{bmatrix} k_u & 0 & 0 & 0 \\ 0 & k_u & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Finally, use M_2 to compute the transformation of the previously chosen world point and call the result p_2 .

The angle ψ can now be determined by making use of Equation 1 and the following observation. The only differences between M_1 and M_2 are their focal lengths, a scale factor, and a rotation about the W axis. Although the scale factor is unknown, it is equal in the U and V directions because this was compensated for in computing M_2 . Together the scale factor and focal length differences serve only to change the size of the image and impose no other distortions. Observe that p_1 is the image point that would be obtained if there were no rotation about the W axis, no scaling of the image, and no cropping of the image. Similarly, p_2 is the image point that is obtained by starting with the true image point associated with the chosen object point and undoing the effects of cropping and unbalanced scaling. Any difference between p_1 and p_2 must be the result of different-sized images

or of rotation about the W axis. Since this rotation is centered about the origin of this coordinate space, the angle ψ can be determined by measuring the angle between p_1 and p_2 at the origin. The differing focal lengths and scale factors can affect only the distance of the points from the origin and cannot alter the angle between the points when measured from the origin.

4 Discussion

The method presented here provides a straightforward way of determining the parameters of the imaging process from a homogeneous coordinate transformation matrix. The geometric interpretation provides some insight into what the equations mean and when they may fail. The appendix illustrates application of the technique to several sets of real data.

In practice, we must be concerned with the robustness of such an algorithm and how it is affected by errors in the data. For example, if f or k are very large, the view angle subtended by the image is small and the projection is nearly orthogonal. In this case, the method becomes sensitive to the precision of the matrix, and only the camera's orientation can be ascertained with confidence. This property is intrinsic in the problem formulation, and any method that derives camera parameters from the correspondence between image and world coordinates is subject to this sensitivity. The parameters computed by the methods outlined in this paper can be used to reconstruct the camera transformation matrix (within a choice of focal length) when synthetic data are used. When empirical data are used, as in the appendix, instabilities in the matrix often make it impossible to reconstruct it with accuracy.

The camera model used throughout this paper is somewhat simplified. The image plane has been assumed to be perpendicular to the principal ray and the image axes are assumed to be perpendicular. Furthermore, the effects of a non-ideal lens and other nonisotropic distortions have been ignored. The accuracy of the decomposition will degrade if these assumptions are not valid. While we do not expect this technique to be any more robust than that of Ganapathy, we do feel that its geometric interpretation provides useful clues as to when it will be dependable. The method's utility has been demonstrated on actual photographs and because it is non-iterative, the computational burden is insignificant.

5 Acknowledgments

I would like to thank Robert Bolles for his comments and patience in debugging the software for least-squares determination of a camera transformation matrix, and for supplying that program in the first place. I am also grateful for discussions with Kicha Ganapathy as to the relation between this method and his own.

Bibliography

- [1] Ballard, D. H., and Brown, C. M., *Computer Vision*, Prentice-Hall, New Jersey, 1982.
- [2] Cameron, R., *Above San Francisco*, Cameron and Company, San Francisco, 1976.

- [3] Fischler, M. A., and Bolles, R. C., "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography", *Communications of the ACM*, Vol. 24, No. 6, June 1981, pp. 381-395.
- [4] Ganapathy, S., "Decomposition of Transformation Matrices for Robot Vision", *IEEE*, 1984, pp. 130-139.
- [5] Gennery, D. B., "Stereo-Camera Calibration", *Proceedings of the Image Understanding Workshop*, November 1979, pp. 101-107.
- [6] Lowe, D. G., "Solving for the Parameters of Object Models from Image Descriptions", *IU Workshop*, April 1980, pp. 121-127.
- [7] Roberts, L. G., "Machine Perception of Three-Dimensional Solids", MIT Lincoln Lab, Technical Report No. 315, May 1963.
- [8] Rogers, D. F., and Adams, J. A., *Mathematical Elements for Computer Graphics*, McGraw-Hill Book Company, New York, 1976.
- [9] Sobel, I., "On Calibrating Computer Controlled Cameras for Perceiving 3-D Scenes", *Artificial Intelligence* 5, 1974, pp. 185-198.
- [10] Sutherland, I. E., "Three Dimensional Data Input by Tablet", *Proceedings of the IEEE*, Vol. 62, No. 4, April 1974, pp. 453-461.
- [11] Thompson, Morris M., *Manual of Photogrammetry*, American Society of Photogrammetry, Falls Church, Virginia, 1966.
- [12] Ullman, S., *The Interpretation of Visual Motion*, The MIT Press, Cambridge, Massachusetts, 1979.

A Examples

We now present two examples to illustrate our technique.

A.1 Imagery from Robotics Applications

Ganapathy used the following experimentally determined 3×4 matrix to demonstrate his method [4].

$$\begin{bmatrix} -2.3819 & 0.49648 & -.039462 & 847.40 \\ -.043897 & -.062872 & -2.4071 & 882.91 \\ -.00026388 & -.00062759 & -.000071843 & 1.0 \end{bmatrix}$$

This matrix is used to obtain the image coordinates (u, v, s) by premultiplying the world coordinates, (x, y, z, t), by the matrix. To make it compatible with the notation used throughout this paper, it must be transposed and an arbitrary column vector inserted. This column is the one that determines w and does not affect the imaging process. The matrix, suitably rewritten, is

$$M_1 = \begin{bmatrix} -2.3819 & -.043897 & 0.0 & -.00026388 \\ 0.49648 & -.062872 & 0.0 & -.00062759 \\ -.039462 & -2.4071 & 1.0 & -.000071843 \\ 847.40 & 882.91 & 0.0 & 1.0 \end{bmatrix}$$

From this matrix the following values were obtained by the method presented in Section 3.

$$L = (620.51, 1295.68, 321.64)$$

This agrees closely with Ganapathy's determination of the camera's location:

$$(620.9344, 1295.476, 321.8140)$$

The orientation of the camera is computed from the normal to the image plane:

$$\vec{n} = (-.3855, -.9167, -.1049)$$

This yields (in degrees)

$$\theta = 157.1949 \quad \phi = -6.0239 \quad \psi = 359.909$$

in agreement with Ganapathy's results:

$$\theta = 157.1951 \quad \phi = -6.023912 \quad \psi = 359.6915$$

Other parameters obtained from M_1 include

piercing point: $(u_0, v_0) = (682, 478)$ in pixels

focal length: $f \cdot k_u = 3488$

$$f \cdot k_v = 3485$$

relative scale factor: $k_u/k_v = 1.0009$

A.2 Outdoor Imagery

Figure 8 shows a photograph taken from a book of pictures of San Francisco [2]. It was necessary to determine the imaging geometry in order to use the picture for work in image understanding. Ground truth data were obtained manually from a map of the city. A total of fifteen pairs of image and world coordinates were used to obtain the following camera transformation matrix with a least-squares program:

$$M_2 = \begin{bmatrix} .172137 & .131132 & 0.0 & .000346452 \\ -.15879 & .112747 & 0.0 & .000311253 \\ .0187902 & .291494 & 1.27976 & .0000656643 \\ 274.943 & 258.686 & 0.0 & 1.0 \end{bmatrix}$$

The results computed from this image are plotted on the map in Figure 9 and described further below. The camera location was computed to be near the intersection of California and Mason Streets at an elevation of 435 feet above sea level. The camera was oriented as shown on the map, at an angle of 8° above the horizon. Computation of the piercing point is sensitive to errors in the matrix because the projection is nearly orthographic, but the location derived is marked by the point P in the image in Figure 8. The focal length and scale factor relations were computed to be $f \cdot k_u = 495$ and $f \cdot k_v = 560$, indicating an aspect ratio of $k_u/k_v = .88$.

Figures 10 and 11 show the results for another photograph of San Francisco. The camera transformation matrix computed from 16 points of ground truth data is shown here:

$$M_3 = \begin{bmatrix} -.175451 & .0269801 & 0.0 & .000151628 \\ -.105205 & -.0963531 & 0.0 & -.00016085 \\ .0043556 & .23031 & 1.07834 & .0000159749 \\ 297.836 & 249.574 & 0.0 & 1.0 \end{bmatrix}$$

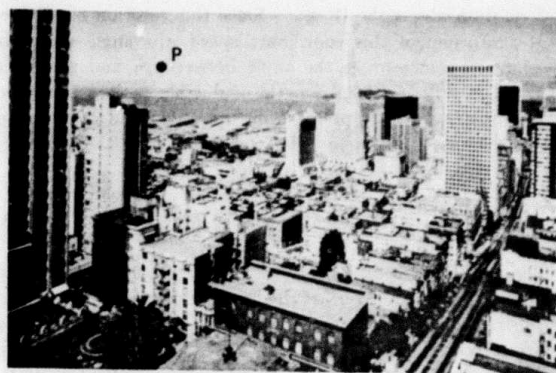


FIGURE 8 PHOTOGRAPH OF SAN FRANCISCO

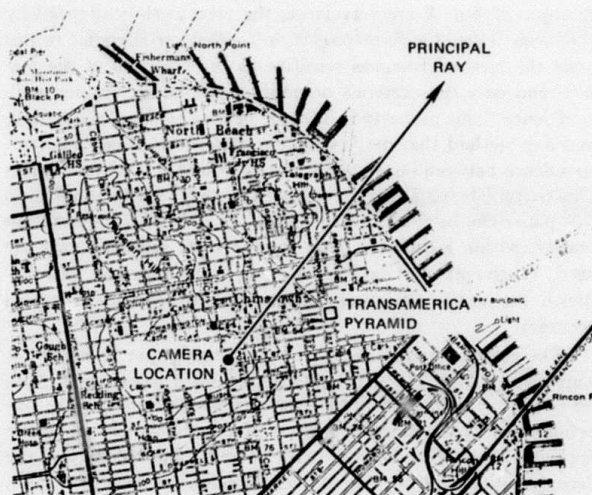


FIGURE 9 MAP OF SAN FRANCISCO

The elevation of the camera was found to be 1200 feet above sea level and the inclination was 4° above the horizon. The horizontal location and orientation are plotted in Figure 11. The piercing point was unreliably computed to be at the point P in Figure 10. The focal length and scale factor relations were $f \cdot k_u = 876$ and $f \cdot k_v = 999$, implying an aspect ratio of $k_u/k_v = .88$.

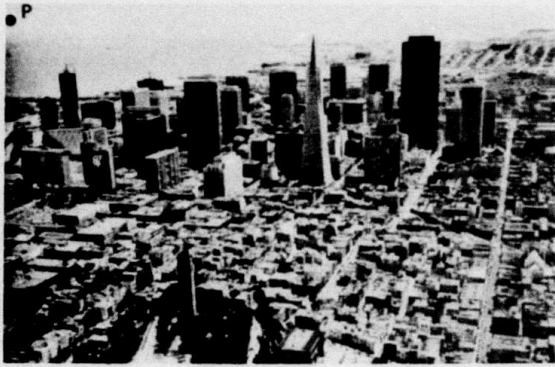


FIGURE 10 ANOTHER PHOTOGRAPH OF SAN FRANCISCO

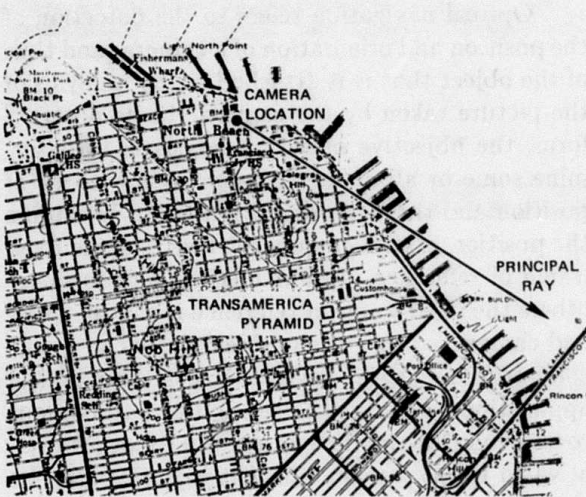


FIGURE 11 MAP OF SAN FRANCISCO

Optical Navigation by the Method of Differences

Bruce D. Lucas and Takeo Kanade
Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract. The *method of differences* refers to a technique for image matching that uses the intensity gradient of the image to iteratively improve the match between the two images. Used in an iterative scheme combined with image smoothing, the method exhibits good accuracy and a wide convergence range. In this paper we show how the technique can be used to directly solve for the parameters relating two cameras viewing the same scene. The resulting algorithm can be used for optical navigation, which has applications in robot arm guidance and autonomous roving vehicle navigation. Because of the regular structure of the algorithm, the prospects of carrying it out with special-purpose hardware for real-time control of a robot seem good. We present experimental results demonstrating the accuracy and range of convergence that can be expected from the algorithm.

This research was sponsored in part by the Defense Advanced Research Projects Agency (DOD), ARPA order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539, and in part by the Office of Naval Research under Contract N00014-81-K-0503.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring agencies or of the US Government.

1. Introduction

Optical navigation refers to the detection of the position and orientation of a camera, and thus of the object that it is attached to, by analysis of the picture taken by the camera. In its abstract form, the objective of such analysis is to determine some or all of the six parameters (three of position and three of orientation) that determine the position of that camera relative to some fixed frame of reference. In our method and in many others the fixed frame of reference is that of a second camera, so that the problem is that of image comparison. Optical navigation has a number of applications including navigation of autonomous roving vehicles and navigation of a robot arm relative to the object on which it is performing its task. We will discuss these applications in more detail below.

Several approaches to optical navigation are possible. These may be divided into three categories: sparse two-dimensional matching, continuous two-dimensional matching, and three-dimensional matching. In the sparse two-dimensional approach, one starts with a discrete set of points in the two images that match, and deduces the camera motion. In the case of a sparse set of points, there is a question of how many points are necessary to uniquely solve for the camera parameters; this has been addressed by Tsai & Huang (1981). With more points, the problem is overspecified and a least-squares approach is required (Gennery, 1980). In the continuous two-dimensional matching approach, one

starts with a whole image field of matches (the "optical flow field"); Bruss & Horn (1983) have shown how to determine the camera motion from the optical flow field, again using a least-squares formulation. Obtaining the optical flow field has been investigated by several people, for example Horn & Schunck (1981) and Cornelius & Kanade (1983). In the three-dimensional matching approach, corresponding points in three dimensions (obtained e.g. by stereo) are used to determine the camera motion; this technique was used by Moravec (1980) to navigate a rover.

These approaches all split the process into two steps: finding the matches and using those matches to solve for the camera parameters. In this paper we show how to combine the two steps into one, by applying a generalized image matching technique that we term the *method of differences*. The method of differences directly computes the six camera parameters, much as standard matching techniques compute two parameters (the x and y displacements). That is, the camera parameters are explicitly included in the matching process. The method takes advantage of the fact that, in many applications the approximate position and orientation of the camera are known. Starting from that estimate we compute a better estimate by using the image intensity gradient as a guide. By using an iterative scheme our estimates converge to the correct value. The result is a technique that is fast and free of search.

The method of differences and similar methods based on image gradients have been applied before to the analysis of small image motions (Limb & Murphy, 1975; Cafforio & Rocca, 1979), to optical flow field determination (Horn & Schunck, 1981), and to stereo image analysis (Lucas & Kanade, 1981). Here we show how it can also be applied to navigation.

In the remainder of the paper, we first present several robotics scenarios calling for optical navigation. Then we describe the method of differences in a one-dimensional case, which serves to illustrate many of the issues. Then we show how the same technique can be used for multi-parameter estimation. Finally, we present

some experimental results and draw some conclusions.

2. Applications

Many robotic tasks require a knowledge of the position and orientation of the "robot." This is because mechanical imperfections and environmental uncertainty make it impossible to know exactly how a robot will move in response to the commands sent to it and exactly what it will encounter in its surroundings. Optical navigation takes its place alongside various types of mechanical navigation to correct this problem, and indeed has distinct advantages with respect to responding to environmental uncertainty over those methods. Such tasks can be classified along several dimensions; three of particular interest to us are the nature of the robotic agent, the nature of the environment, and the manner in which the optical navigation is used.

Robotic agents. For our purposes, robotic agents can be roughly divided into two classes: fixed-base robot arms and autonomous roving vehicles (although many vehicle designs call for the rover to have a robotic arm or manipulator of some sort). In both cases optical navigation can play an important role, although the nature of the navigation may be different.

For example the "world" in which a manipulator moves is generally smaller than that in which a rover moves, and so the nature and quality of the image obtained may differ (consider such effects as depth of field). Moreover, the two types of agent differ with respect to the availability of other sources of information about the robot's movement. A rover might well be capable of inertial or radio navigation in addition to optical navigation. On the other hand robot arms typically operate in environments where such techniques as structured lighting may feasibly provide additional information, while such is generally not the case for rovers. Such additional sources of knowledge could be incorporated into the method, for example to provide the necessary initial estimate of position.

A second point of difference is that the mechanisms for control of an arm and a rover are quite different: an arm is generally controlled by the coordinated rotations of a number of joints, while the a rover is moved under the power of a number of wheels or legs. Thus, the method must be adapted differently in each case if it is desired to directly solve for the joint positions, the wheel rotations, or even the control signals that cause those movements.

Another point of difference is in the degree of constraint that exists in the motions of the robot. Typically, a manipulator will be able to move in all six degrees of freedom, while a rover may only have freedom in say the pan, x , and z motions. These constraints bear on the specific formulations of the technique for the different tasks.

Environment. In a known environment, the robot must perform a series of maneuvers with respect to a set of objects in the environment whose nature and approximate position are known in advance. In an unknown environment case, the robot must be prepared to encounter anything, or at least a variety of objects. Tasks in which a robot arm must operate in an unknown environment are conceivable, but unlikely; therefore we will confine ourselves to the rover case, but our remarks concerning the rover operating in a known environment will apply equally well to the robot arm case.

In the case of a known environment, the scenario for the use of the method of differences is as follows: the robot is taken through the series of operations that it will be expected to perform. As it does this it records a number of camera views sufficient to cover the substantially different situations the robot will encounter. These images will serve as landmarks with respect to which the robot will later navigate. Then a number of reference points are chosen from each image and are assigned a distance from the camera. This can be done a number of ways: a second camera in a known position with respect to the first can provide a stereo baseline for making the measurements; a known model of the environment can be fitted to those points; the dis-

tances can be directly measured; the distances can be obtained by structured lighting; and so on. In any case, since this is a training step to be done only once, the assignment of distances need not be completely automated. This concludes the training process. Then, at each step of an actual run, the image received by the camera will be compared against one of the stored reference images (actually, only the positions and intensities of the reference points are need be stored), and the method of differences will be used to determine the position of the robot relative to the reference coordinate system. In a variation of this process, the method of differences can be used to directly solve for the control signals to be fed to the robot.

The case of an unknown environment is more difficult. It is not possible to store reference images, so the process of selecting and determining the distances of reference points must be carried out anew each time the rover encounters a new situation. Techniques for doing this, such as by automatic stereo vision, are themselves the subject of research. Once the reference points are located and their distances determined, they can be used to navigate until they are lost from view. If necessary, the stereo solver can be called again at each step to refine the estimates of the distances and to determine the distances of new reference points acquired to replace old ones that have disappeared from view.

Manner of application. Finally, optical navigation can be used in a robot in one of two ways. If the position of the robot can be calculated quickly enough (for example with the aid of special-purpose hardware), then the result can be used in a continuous feedback loop. Two aspects of the method are favorable for this mode: in a feedback loop the range of convergence of the algorithm need not be large, because the position will be calculated frequently enough that the robot will have moved relatively little during that time. Moreover, the method need not accurately compute the position of the robot, but rather needs to be taught what the camera should see when it is in the desired position; as long as

the method is able to provide signals to move in the proper direction when the camera is out of position and provided that the method can detect when the camera is seeing what it is expected to see, the position of the camera will converge to the correct position. Of course proper precautions are necessary to prevent oscillation.

Even if the method cannot be applied fast enough to be used in a feedback mode, it can still be fruitfully used in a "stop-and-go" mode. In such a case the rover will move greater distances between each application of the method, and so the demands on the optical navigation are greater: it must exhibit in this case a greater range over which it will converge to the proper value.

3. The technique

Parameter estimation by the method of differences. First we present the technique for the one-dimensional, one-parameter case to give its flavor. Consider two one-dimensional images $I_1(x)$ and $I_2(x)$ related by a translation, so that $I_1(x) = I_2(x + h)$; we wish to estimate the translation h . One way to do this is to find that \hat{h} that minimizes the total squared error,

$$E = \sum_x (I_2(x + \hat{h}) - I_1(x))^2.$$

Since we want a local, non-searching algorithm, we approximate $I_2(x + \hat{h})$ using $I_2(x)$ on the basis of local information, namely the derivative; this yields the approximation

$$E \approx \sum_x (I_2(x) + \hat{h} D_x I_2(x) - I_1(x))^2, \quad (1)$$

where D_x denotes partial differentiation with respect to x . This equation is quadratic in \hat{h} , so we can differentiate with respect to \hat{h} , set equal to zero, and solve the resulting linear equation for \hat{h} , obtaining

$$\hat{h} = \frac{\sum_x (I_1(x) - I_2(x)) D_x I_2(x)}{\sum_x D_x I_2(x)^2}. \quad (2)$$

This one-parameter case illustrates the nature of the method. We call it the method of differences because it is based on comparing the dif-

ference between the images, $I_1(x) - I_2(x)$, with the derivative $D_x I_2(x)$ (which will in fact be implemented as a difference), to obtain an estimate for the parameter h .

Iteration and smoothing. As it stands, this method would not work very well. However, two additional modifications make it a viable technique. First, because the method yields only an approximation \hat{h} to the disparity h , we must use an iterative scheme to obtain an accurate result. The idea is to calculate an estimated disparity, move I_2 by that amount, and calculate again. Starting with an initial estimate \hat{h}_0 , the iteration is given by

$$\hat{h}_{i+1} = \hat{h}_i + \frac{\sum_x (I_1(x) - I_2(x + \hat{h}_i)) D_x I_2(x + \hat{h}_i)}{\sum_x D_x I_2(x + \hat{h}_i)^2}. \quad (3)$$

Note that if $\hat{h}_i = h$, then our calculated increment is zero (since $I_1(x) = I_2(x + h)$), so h is indeed a convergence value for the algorithm.

Second, to improve the accuracy and range of validity of the linear estimate used in (1), we must smooth the image. This can be thought of as smoothing out purely local lumps and wrinkles in the image intensity profile that would make a linear estimate accurate only over a small range. Alternatively, smoothing can be understood in the frequency domain as reducing the high frequency components. We have shown (Lucas, 1984) that \hat{h} as calculated in (2) can be expressed as

$$\hat{h} = \frac{\sum_{k \geq 0} r_k^2 k \sin 2\pi k h}{2\pi \sum_{k \geq 0} r_k^2 k^2}, \quad (4)$$

where the r_k are the coefficients of the Fourier series for I_2 :

$$I_2(x) = \sum_{k \geq 0} r_k \sin(x + \theta_k).$$

This is a very interesting result, because it says that the estimated disparity \hat{h} depends only on the power spectrum τ_k^2 of the image, which tends to be similar from image to image: and is independent of the phase spectrum, which is where all the information in the image is (Kretzmer, 1952; Oppenheim & Lim, 1981). In particular, the power spectrum τ_k^2 tends to fall off as k goes to infinity. Thus the behavior of the disparity calculation in (2) will be similar from image to image, and so experiments can yield generally valid results.

Indeed, inspection of (4) shows that the estimated disparity \hat{h} , as a function of the real disparity h , is zero when h is zero (as already noted), increases as h increases (with the proper sign), and finally falls back to zero as h increases further. (Examples of such curves are presented later in Figure 3). Furthermore, the more high-frequency terms are present, the smaller are the values of disparity h for which the disparity estimate \hat{h} falls off to zero. For example, if $I_1(x)$ were a pure sine wave, then we could not in principle calculate the disparity if h is greater than one-half wavelength. Both lines of reasoning thus lead to the conclusion that smoothing will extend the range disparities h over which the iteration will converge. While this analysis is not exact (for example, (4) is strictly true only if the sum in (2) extends over the whole real line or over one period of a periodic function), experiments we present below will verify the applicability of the conclusion.

Thus, larger smoothing windows yield wider disparity ranges over which the algorithm converges, but may not produce an accurate answer. The relationship between iteration and smoothing is that successive steps of the iteration can be done with progressively less smoothed images, in a sort of coarse-fine approach. This allows the algorithm to tolerate a large disparity yet yield an accurate answer.

Multi-parameter case. The multi-parameter case is analogous. Consider the camera model illustrated in Figure 1. We start with

an estimate of the vector c of six camera parameters, which define the relationship between the two camera coordinate systems. We wish to estimate the effect that a change Δc in the camera parameters c will have on the error, given by

$$E = \sum_{\mathbf{p}} (I_2(\mathbf{v}) - I_1(\mathbf{p}))^2$$

After the change, the error will be approximately

$$E \approx \sum_{\mathbf{p}} (I_2(\mathbf{v}) + \Delta c (D_c \mathbf{u})(D_u \mathbf{v})(D_v I_2(\mathbf{v}) - I_1(\mathbf{p}))^2. (5)$$

Here, $D_c \mathbf{u}$ indicates the matrix of partial derivatives that describes how the position of the three-space point \mathbf{u} in the Camera 2 coordinate system varies as the position of that system (which is determined by c) is varied; $D_u \mathbf{v}$ describes how the projection \mathbf{v} of the point \mathbf{u} onto the Camera 2 coordinate system varies as the position of that point varies; and $D_v I_2$ is simply the intensity gradient of the image I_2 . (We use row vectors for c etc. so that D_c etc. can be prefix operators; D_c for example is a column vector of the partial derivatives w.r.t. c .) Their product, by the chain rule, tells us how the image intensity of I_2 at the presumed match \mathbf{v} to the point \mathbf{p} varies as we vary the camera parameters c . By knowing this at each of the many points \mathbf{p} over which the summation runs, we can estimate how we should change the camera parameters c to make the intensities at each of the points \mathbf{v} match those at their respective matching points \mathbf{p} . To do this, we differentiate (5) with respect to Δc , set equal to zero, and solve obtaining

$$\Delta c = \left[- \sum_{\mathbf{p}} (I_2(\mathbf{v}) - I_1(\mathbf{p})) (D_c I_2(\mathbf{v})) \right] \cdot \left[\sum_{\mathbf{p}} (D_c I_2(\mathbf{v})) (D_c I_2(\mathbf{v}))^T \right]^{-1}, (6)$$

where

$$D_c I_2(\mathbf{v}) = (D_c \mathbf{u})(D_u \mathbf{v})(D_v I_2(\mathbf{v})).$$

Note that $D_c I_2(\mathbf{v})$ is a 6×1 matrix (column vector), so that the second sum of (6) is a matrix

that must be inverted. This raises the question of under what conditions this inversion will be possible. We have shown elsewhere (Lucas, 1984) that this matrix will be singular under orthography in the case where all six parameters are to be solved for, and conversely that it will be non-singular under perspective. However, if the reference points are positioned in three-space so that the perspective projection is well-approximated by the orthographic projection, then the matrix will be nearly singular. This implies that the points should not be too far from the camera and should be well-distributed in distance from the camera.

Having computed Δc according to (6), we update our estimate c to $c + \Delta c$, yielding a better estimate. This procedure results in an iterative scheme analogous to that in (3). Moreover, the previous comments concerning smoothing the images to improve the range of convergence apply as well to the multi-parameter case.

4. Experimental results

Our experimental data consisted of three views of the same scene taken by a camera mounted on the Stanford cart (Moravec, 1980); they are shown in Figure 2. The camera was mounted on a slider, so we had accurate knowledge of the relative positions of the cameras. The three views were pictures taken by the camera at the left, middle, and right slider positions, with 26 cm separating each position. The left picture was used as the reference image, and a number of points were selected from this image as reference points. These are the points p that the sum in (5) runs over. Then the right picture was used as the second image of a stereo pair to obtain the distances $z(p)$ of the reference points p . The method was then used to determine position of the middle camera. Since the position of the middle camera was known, we could assess the accuracy of the method. Moreover, by varying around the correct value the initial estimate of the middle camera's position that we provided to the algorithm, we could determine the range of convergence.

Convergence range. The convergence range in a typical one-parameter case can be determined from the graph in Figure 3. In this experiment we solved for x while holding the other five parameters fixed. This graph illustrates that the computed adjustment (vertical axis) to the x estimate (horizontal axis) is zero when the value of x is at the convergence point near the correct value of 26 cm, increases as the disparity increases and has the proper sign, and then falls back to zero. The range of convergence is the interval over which the computed adjustment is, say, at least 0.1 times that required to move the estimate to the correct answer at the point where the curves cross the horizontal axis, which is roughly the same as the region over which the adjustment has the right sign. For the largest smoothing window, the range of convergence is nearly a meter in one direction and well over a meter in the other direction; the range is smaller for the smaller smoothing windows. Note that the convergence value for the larger window is not the same as for the smaller windows, but is well within their convergence range. This means that a coarse-fine technique would work, and indeed could skip right from the largest window to a much smaller one, avoiding the need to calculate a multitude of smoothed images. The convergence range for y is similar, and for z is somewhat larger (although the result is less accurate). The convergence ranges for pan and tilt are ± 10 degrees or so, and the convergence range for roll is around ± 30 degrees. These convergence ranges (except for roll) are of course dependent on the geometry of the situation.

What is the relationship between these convergence ranges and the convergence ranges in the multi-parameter case? This is shown in Figure 4. We see that if we solve for two parameters (pan and tilt, top graph), the range is smaller than the range that would be expected on the basis of the one-parameter results for pan and tilt alone; and if we solve for all six parameters (bottom graph), it is smaller still. Nevertheless, the range is still quite adequate for the continuous feedback mode. Whether it is adequate for the stop-and-go mode,

which involves a larger motion at each step, depends on the accuracy of the arm and on the accuracy of other navigational aids that can provide the initial estimates.

Accuracy. To assess the accuracy under a variety of conditions, we select reference points using a variety of methods, including by hand and by computer, resulting in several sets of data points of various sizes. Then we doubled the number of sets of reference points by either applying or not applying a pruning process to the sets we had. This pruning process, which is described elsewhere (Lucas, 1984), was based on the method of differences and served to improve the accuracy of the stereo matches. It also eliminated some points as being unfit for use by the method, for example because they were in a region of small gradient. The results are shown in Figure 5. Several general trends are observable. First, using more points produces more accurate results. Second, the pruning process can improve the results, as evidenced by the left endpoints of the lines in the figure being lower than the right endpoints. These two factors are of course in conflict, and the improvement due to the pruning process is apparent only provided the number of points is not reduced too much. Finally, the accuracy does not seem to be affected much by the number of parameters solved for.

Implementation. The implementation may be divided into two parts: smoothing and camera parameter estimation. The smoothing must be done over a relatively large window, up to 65×65 in our experiments. It is the most time-consuming, even though we implemented it as uniform smoothing over a rectangular region, which by a well-known algorithm takes a constant number of operations (two additions and two subtractions) per pixel, regardless of the size of the smoothing window. However, it is fairly well understood how to build special-purpose hardware for doing smoothing quickly, essentially in real time.

The parameter estimation step is more interesting. Our implementation, in which no attention was paid to efficiency, requires approximately 3 to 4 ms per reference point per iteration on a VAX 11/780. In the continuous feedback mode, only one iteration per time step would be used since only an approximate answer is needed. Thus 50 reference points (the largest number used in the experiments reported above) would require less than 200 ms per time step. This figure could probably be improved severalfold by more careful coding and taking account of the fact that some of the entries in the matrices in equation (6) are known *a priori* to be zero. This information, together with the fact that the algorithm has a regular structure free of decision points that could easily be implemented in special-purpose hardware, suggests that it is feasible for real-time control of a robot.

It should be noted that a considerable storage savings is possible with respect to the reference image. The image intensities of the reference image and its smoothed versions are needed only at the reference points p . Thus, the entire reference image need not be stored.

We have demonstrated that the method of differences provides a useful technique for optical navigation. We have shown that the algorithm can successfully determine all six camera parameters. It converges to the correct position given an estimate within something on the order of a meter (less if more parameters are solved for), and converges to a result accurate to a centimeter or so (regardless of the number of parameters solved for). Moreover, it can do so using 50 or less reference points. Because of the regular structure of the algorithm, the prospects of carrying out the calculations in real time with special-purpose hardware seem good.

6. References

A. R. Bruss and B. K. P. Horn, 1983. Passive navigation. *Computer Vision, Graphics, and Image Processing*, 21, 3-20.

C. Cafforio and F. Rocca, 1979. Tracking moving objects in television images. *Signal Processing*, **1**, 133-140.

N. H. Cornelius and T. Kanade, 1983. Adapting optical-flow to measure object motion in reflectance and x-ray image sequences. Proc. ACM SIGGRAPH/SIGART Workshop on Motion: Representation and Perception, Toronto, 50-58.

D. B. Gennery, 1980. Modeling the environment of an exploring vehicle by means of stereo vision. PhD Thesis, Department of Computer Science, Stanford University.

B. K. P. Horn and B. G. Schunck, 1981. Determining optical flow. *Artificial Intelligence*, **17**, 185-202.

E. R. Kretzmer, 1952. Statistics of television signals. *Bell System Tech. J.*, **31**, 751-763.

J. O. Limb and J. A. Murphy, 1975. Estimating the velocity of moving images in television signals. *Computer Graphics and Image Processing*, **4**, 311-327.

B. D. Lucas, 1984. Generalized image matching by the method of differences: algorithms and applications. PhD Thesis (in preparation), Computer Science Department, Carnegie-Mellon University.

B. D. Lucas and T. Kanade, 1981. An iterative image registration technique with an application to stereo vision. Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver.

H. P. Moravec, 1980. Obstacle avoidance and navigation in the real world by a seeing robot rover. Tech. Rept. CMU-RI-TR-3, Robotics Institute, Carnegie-Mellon University.

A. V. Oppenheim and J. S. Lim, 1981. The importance of phase in signals. *Proc. IEEE*, **69**, 529-541.

R. Y. Tsai and T. S. Huang, 1981. Uniqueness and estimation of 3-D motion parameters of rigid objects with curved surfaces. Proc. IEEE Conference on Pattern Recognition and Image Processing.

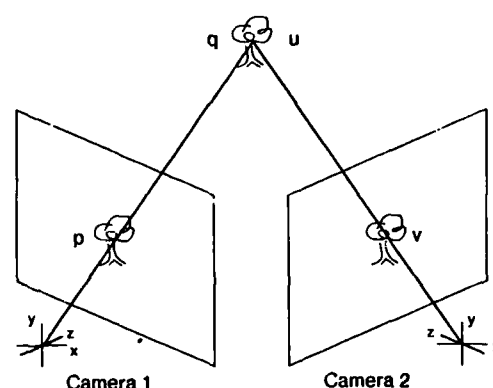


Figure 1. Camera model for optical navigation. Camera 1 defines the reference picture and coordinate system, with the origin at the "pinhole." For any point $p = [p_x \ p_y]$ in the reference image there is a point $q = [q_x \ q_y \ q_z]$ in three-space that produced the image, at depth $z(p) = q_z$. This point is expressed as $u = [u_x \ u_y \ u_z]$ in the Camera 2, or test, coordinate system. The relationship between q and u is a function parameterized by the six camera parameters c : three for the relative positions of the cameras and three for their relative orientation. Finally, the three-space point appears at the point $v = [v_x \ v_y]$ in the Camera 2 image plane. The points p and v are said to correspond.

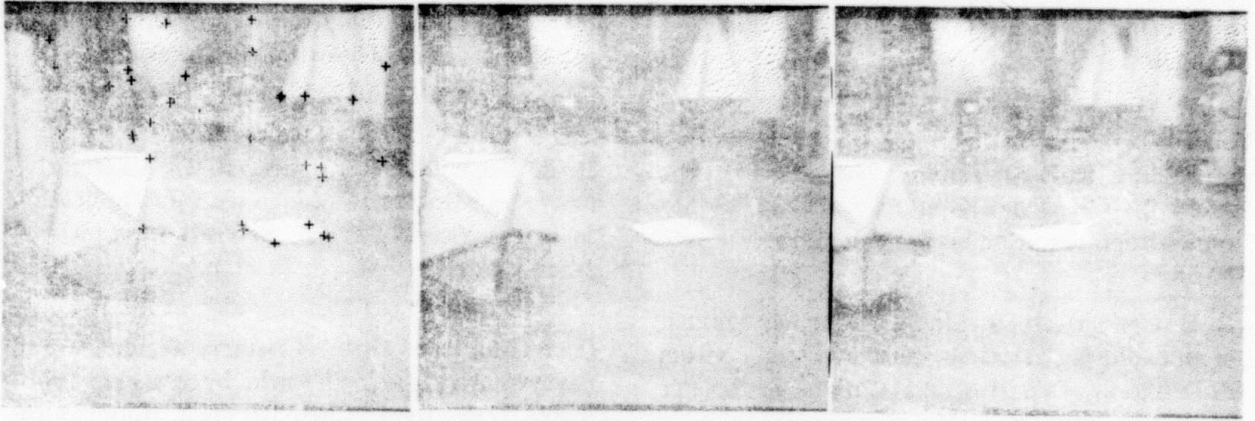


Figure 2. Experimental data. Left, middle, and right views of the same scene. Reference points are shown on left (reference) image.

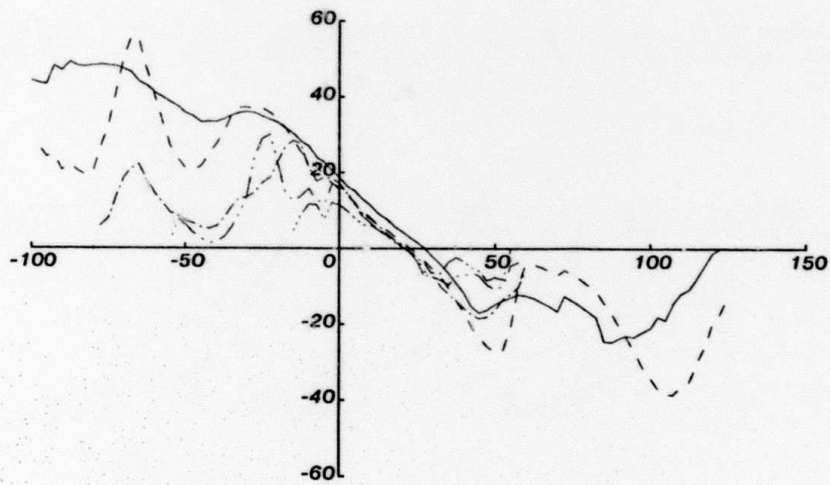


Figure 3. Behavior of algorithm in one-parameter case. Horizontal axis represents initial estimate of x provided, vertical axis represents computed Δx , both in cm. Solid curve represents largest smoothing window, 65×65 ; others represent smaller windows.

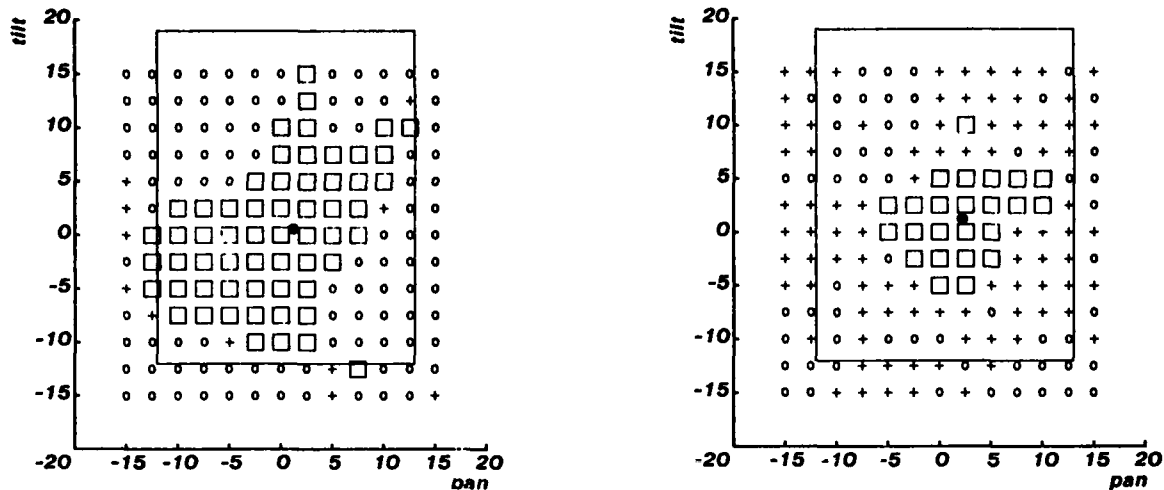


Figure 4. Left graph shows, for each initial value of pan and tilt, whether the algorithm converged to the correct value (large boxes), converged to the wrong value (small circles), or failed to converge (pluses). Solid dot is correct value, big rectangle indicates range predicted by single-parameter results. Right graph is a two-dimensional slice of a similar six-dimensional solid, in which all six parameters were solved for.

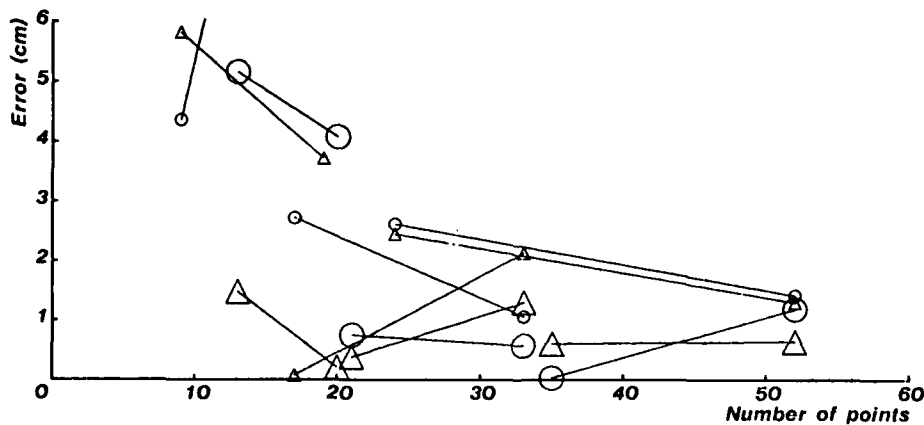


Figure 5. Graph shows the absolute error in x position on images smoothed with 9×9 window. Each point represents the result with a different set of reference points, distinguished by resulting error (in cm) on the vertical axis, and by number of points in the reference set on the horizontal axis. Triangles indicate the case where three parameters were solved for, circles six. The point at the left end of each line represents a reference set in which a pruning process was carried out on the points represented by the right end of the line. Large points represent image pair discussed in text, small points represent a different image pair.

Autonomous Scene Description with Range Imagery

David R. Smith and Takeo Kanade

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

This paper presents a program to produce object-centered three-dimensional descriptions starting from point-wise 3D range data obtained by a light-stripe rangefinder. A careful geometrical analysis shows that contours which appear in light-stripe range images can be classified into eight types, each with different characteristics in occluding vs. occluded and different camera/illuminator relationships. Starting with detecting these contours in the iconic range image, the descriptions are generated moving up the hierarchy of contour, surface, object, to scene. We use conical and cylindrical surfaces as primitives. In this process, we exploit the fact that coherent relationships, such as symmetry, collinearity, and being coaxial, which are present among lower-level elements in the hierarchy allow us to hypothesize upper-level elements. The resultant descriptions are used for matching and recognizing objects. The analysis program has been applied to complex scenes containing cups, pans, and toy shovels.

1. Introduction

The research presented in this paper aims at producing object-centered three-dimensional descriptions starting from point-wise 3D range data obtained by a light stripe range finder. While most of the initial work in range data analysis was on generating object descriptions of simple objects:

representation of snakes and dolls made of cylindrical parts by means of generalized cylinders [1, 8] or representation of polyhedra by planes [11, 9], recent work seems to be more concerned about efficient matching of objects with 3D models [3, 4]. Our emphasis in this paper, however, is data-driven bottom-up autonomous processing, generating object descriptions from complicated scenes without referring to specific pre-stored object models.

Starting from the iconic range data, the descriptions are generated moving up the hierarchy of contour, surface, object, to scene. We use conical and cylindrical surfaces as primitives. While a top-down verification process is important, the bottom-up process for producing plausible, natural, object-level descriptions is at least as crucial in order to realize general vision systems [7, 2] as the task world becomes larger and less contrived. Our approach to the problem is to exploit the fact that coherent relationships, such as symmetry, collinearity, and being coaxial, that are present among lower-level elements in the hierarchy allow us to hypothesize upper-level elements. This is justified because those coherent relationships do not usually occur accidentally [6]. For example, if the same surfaces with the same relationships appear across two scenes, they tend to be grouped into one object; if one cylinder's axis intersects with another one's, like the relationship between the handle and body of a pan, they tend to belong to the same object. These coherencies must be present because they have been inherited through the hierarchy from the scene level down to the iconic range-data level. Our task is to trace and exploit these coherent relationships reversely for autonomous generation of object descriptions.

¹This research was sponsored by the Defense Advanced Research Projects Agency (DARPA), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

We focused our effort on the use of occluding contours, which can be extracted quite reliably from the light-stripe range data. First, contours are extracted, segmented and classified. From the coherencies among them, such as parallelism, surfaces are hypothesized. These are represented as conic surfaces (pipes, cones, and planes). The surface hypotheses are confirmed or refuted by their ability to account for observed surface area. Hypotheses of surface groups are formed by examining coherencies among the verified surfaces, such as axis intersections. Finally, such surface groupings from multiple scenes are compared. If a similar structure repeatedly occurs, it is identified as an object. The succeeding sections of this paper follow these steps in order.

2. Taxonomy of Contours in Light-Stripe Images

The range images for this work were produced by a light stripe rangefinder, which consists of an illuminator projecting a sheet of light into the scene and a camera that detects amount of deflection in the light stripe on each scan line. Triangulation produces surface-point positions in three dimensions. A range image is shown in Figure 1, which is a composite of the camera's views of the light stripes, using every tenth stripe.

The parallax between illuminator and camera makes ranging feasible, but it also causes occlusions which are difficult to interpret. Figure 2 shows the geometry in light-stripe imaging which include a circular object *A* and the background *B*. It explains how contours are generated which

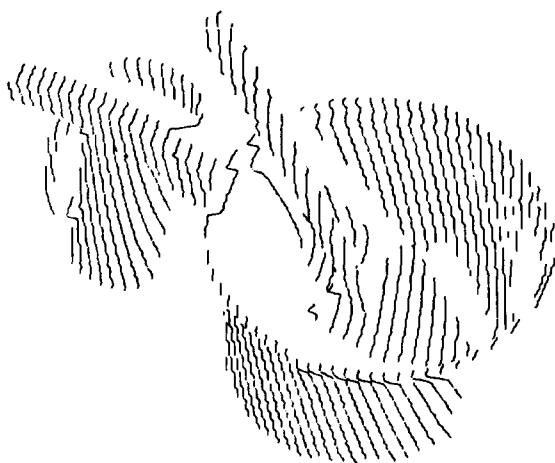


Figure 1: Light-stripe image of a scene

bound a surface against either another surface or a region which cannot be measured. Object *A* cast a deep shadow region *umbra*, which cannot be seen from either the illuminator or the camera. It also casts two half-shadow regions, *penumbras*, which might be seen from either the illuminator or the camera, but not both. No data can be recorded for a surface which lies in either an umbra or a penumbra.

Occlusions occur at the four combinations where the line of sight, either from the illuminator or the camera, is tangential to the surface of the foreground object *A*: that is, *ip1-ip2*, *iul-iu2*, *cp1-cp2*, and *cul-cu2*. Here, the first point of each pair forms the occluding contour and the second the occluded contour; therefore there are eight types of contours in the light-stripe range imagery. This analysis points out a few interesting points. First, previous researchers have dealt only with the simple occlusions involving *cp1-cp2* and *ip1-ip2*, by converting to three dimensions and drawing rays from the illuminator and camera. It can be shown that this can be done without resorting to three-dimensional geometry, by exploiting information in the raw deflection image. Second, interpretation of the occlusions *iul-iu2* and *cul-cu2* provides information about the object, even though the occluding contour is not recorded. This is especially true and useful when we can assume cylindrical objects, because the totally visible region (from *cp1* to *ip1*) is fairly small, and expanding the known part to the region from *iul* to *cul* will greatly increase the accuracy in reconstructing the cylindrical shape.²

Figure 3 shows the contours detected in the image of Figure 1: a) shows all the contour points, b) shows the occluding points, and c) and d) show the penumbral and umbral occluded points, respectively. The occluding contours are directly useful for shape cues, while the occluded contours provide indirect information [10].

²Incidentally, this point suggests that the light-stripe range data be taken with the background, as opposed to the conventional way in which the background is blacked out by a black carpet or curtain.

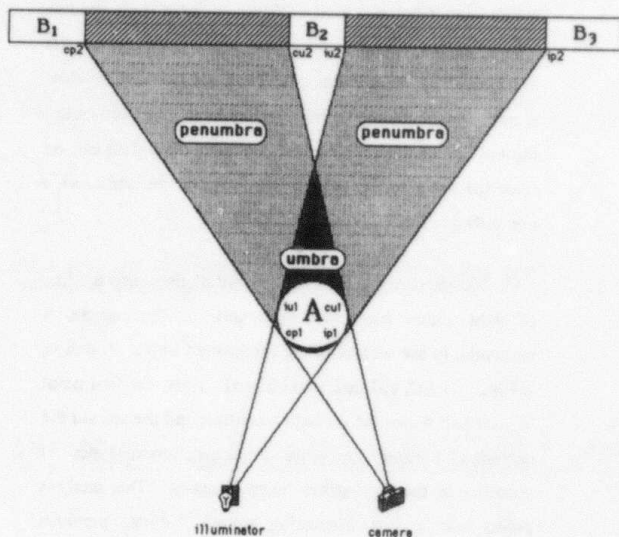


Figure 2: Generation of contours with a light-stripe rangefinder

3. Contour Analysis

3.1. Segmentation

For an extracted contour, local curvatures along it are calculated. The contour is segmented into pieces at the peaks of the curvature value, expecting that the segment between peaks can be described simply. Figure 4 illustrates an example of contour segmentation. The lower part of the figure is a plot of curvature vs. position on the contour, traced clockwise around the shovel handle. In this case, the contour has been divided into three segments at the cross marked positions. The result of segmentation for the whole occluding contours is shown in Figure 5a).

3.2. Segment Classification

Each contour segment is now classified into one of the four shape categories: *straight*, *circle*, *plane*, and *space*. For this, first a straight line is fit by least-squares. If the error residual is low, it is classed as a line. Otherwise, a plane is fit. If the plane does not fit well, the contour is classified as a (nonplanar) space curve. If the plane fits well, a circle is fit to decide whether it should be declared a circle. Figure 5 b) and c) shows the results of classification of the occluding contour segments of our scene.

4. Surface Analysis

Once the contours have been analyzed, they are examined to find what they can tell about the surfaces in the scene. This proceeds in three steps. First, contour segments are grouped (using coherent relationships among them) into ensembles which suggest various primitive surface shapes. Second, specific surface types and equations are proposed for the contour groups together with their spatial extent. Finally, the surface hypotheses are verified in the image data.

4.1. Contour Groups

We seek binary coherence relationships between contours which suggest surface shapes. The relations sought depend upon the shapes of the surfaces, and include:

- Straight vs Straight
 - Parallel
 - Antiparallel
 - Perpendicular
 - Collinear
 - Coplanar
 - Opposed
- Straight vs Circle
 - Perpendicular
 - Stands-up
- Circle vs Circle
 - Parallel
 - Coaxial
 - Concentric
 - Cocircular

Contours sharing appropriate combinations of these relations are aggregated into contour groups. Each contour group suggests a surface, and is classified as one of the following types:

- Cone (including cylinder)
- Ribbon
- Disc
- Plane

As an example, Figure 6 shows what kind of relationships are involved for the group type Cone. The result of the contour grouping for our example scene is shown in Figure 7, whose caption explains the details of individual groups.

4.2. Surface Proposal

Each surface group proposed is converted to a surface equation together with the limits in its spatial extent. Figure 8 shows the surface proposals generated from the corresponding contour groups of Figure 7.

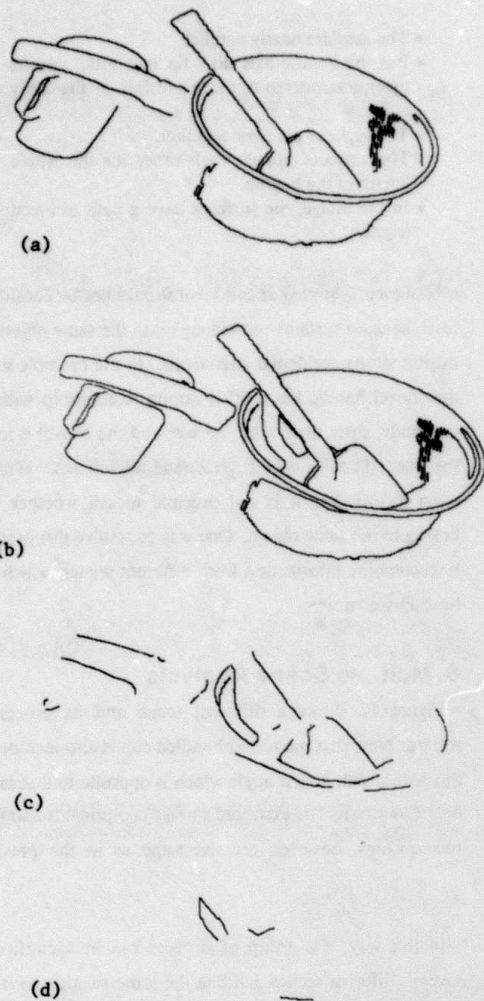


Figure 3: Contours in the image of Figure 1

4.3. Surface Verification

The surface hypotheses are checked by going back to the range image and determining if each hypothesis can account for enough surface area in the scene. This involves a test of position and surface normal in 3-space for each pixel. The pixels which pass this test are summed according to the area which they individually represent in the scene. Based on this test, the proposed surfaces are either accepted or rejected as missing. For example, proposed surface 2 (top of the pan opening) is rejected, because no real surface exists at the proposed position. The surfaces which are accepted are shown together in Figure 9.

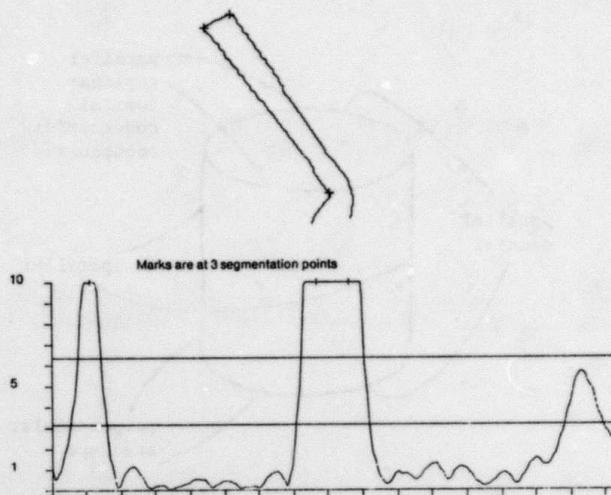


Figure 4: Segmenting a contour. The upper is the contour of the shovel handle, and the graph is a plot of curvature vs position on the contour traced clockwise. The curve is segmented at the tip of the left peak, and on the shoulders of the peak in the middle. The line at 3.2 is the mean curvature for all contour points in the image. The threshold for peak significance is twice the mean (upper line). A more fortuitous choice would have segmented at the right peak, where the handle tapers into the shovel blade at lower right.

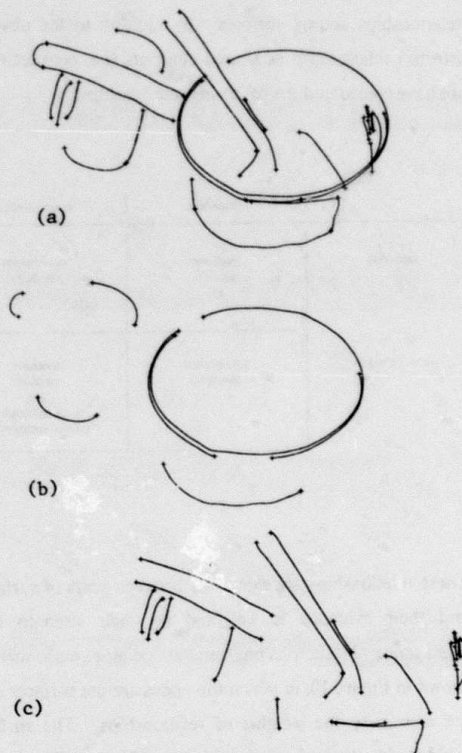


Figure 5: Results of contour segmentation and classification: a) all contours with marks at the segment end points; b) circular contour segments; c) straight contour segments. Short segments are suppressed.

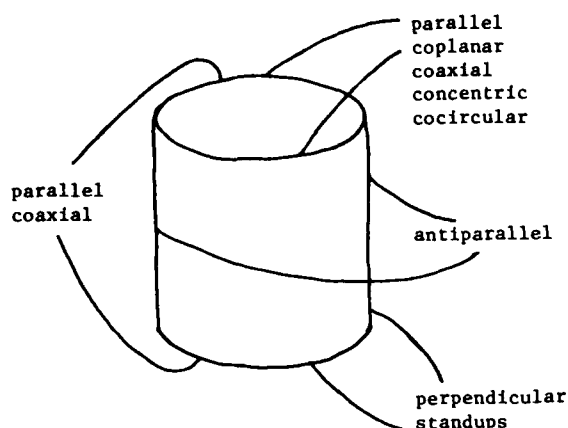


Figure 6: The group type cone and the relationships involved for it. All of the segments and relationships are not necessary to suggest the group

5. Object Analysis

Surfaces are grouped into objects in a similar manner as contour grouping, but in this case based on coherent relationships among surfaces. In addition to the obvious (strong) relationship of shared contours (i.e., connectivity), we have considered the following relationships:

	plane/disc	cone/cylinder
plane/disc	coplanar coaxial gap	axis-in-plane stands-up
cone/cylinder	axis in plane stands-up	coconical coaxial gap intersecting axes interior-interaction

These relationships are examined between pairs of surfaces and their evidence is weighted by their strength and importance. The results are summarized as a graph, such as shown in Figure 10, in which the nodes are the surfaces and the arcs carry the weights of relationships. The surface groups with stronger relationships are shown in Figure 11. For example, the cup body and handle are strongly grouped together because:

- The axes are nearly parallel
- But the handle axis does tip in a little, and its extension intersects the extension of the body axis.
- The surfaces are close together.
- The surfaces oppose each other for the whole length of the handle.
- In the image, the surfaces have a path to bleed together.

However, grouping at this level should not be considered final, because surfaces not belonging to the same object can exhibit strong accidental alignments. In the example scene, the shovel handle has a fairly strong relationship with the pan body, since their axes intersect and the shovel is inside the pan. These are both accidental alignments. Without other knowledge, it is not possible to tell whether they belong to the same object. One way to resolve this problem is to combine information from different scenes, which will be discussed next.

6. Multiple Scene Analysis

Figure 12 shows a different scene and its processing results. Note that the round-handled cup is also included in this scene, with a view angle which is opposite to that in the previous scene. The extracted surface components and their relationships, however, are the same as in the previous scene.

In this way, if a group of surfaces can be identified in several different scenes, holding the same relative positions and orientations, one can assert that they are part of a common object. Relating objects from different scenes is a matching problem, illustrated in Figure 13. Suppose object 1 comprising surfaces *a* and *b* from scene 1 matches object 2 comprising *c* and *d* from scene 2. Then two conditions must be met:

- A surface from object 1 matches a surface from object 2. This is called the *surface match*.
- If the surfaces *a* and *b* are to match surfaces *c* and *d* respectively, then the placement and orientation of *b* with respect to *a* must match the placement and orientation of *d* with respect to *c*. This is called the *transform match*.

The transform match gets its name from the transform which maps the local coordinate system of *b* into the coordinate system of *a*. This must match the transform mapping the coordinate system of *d* into that of *c*.

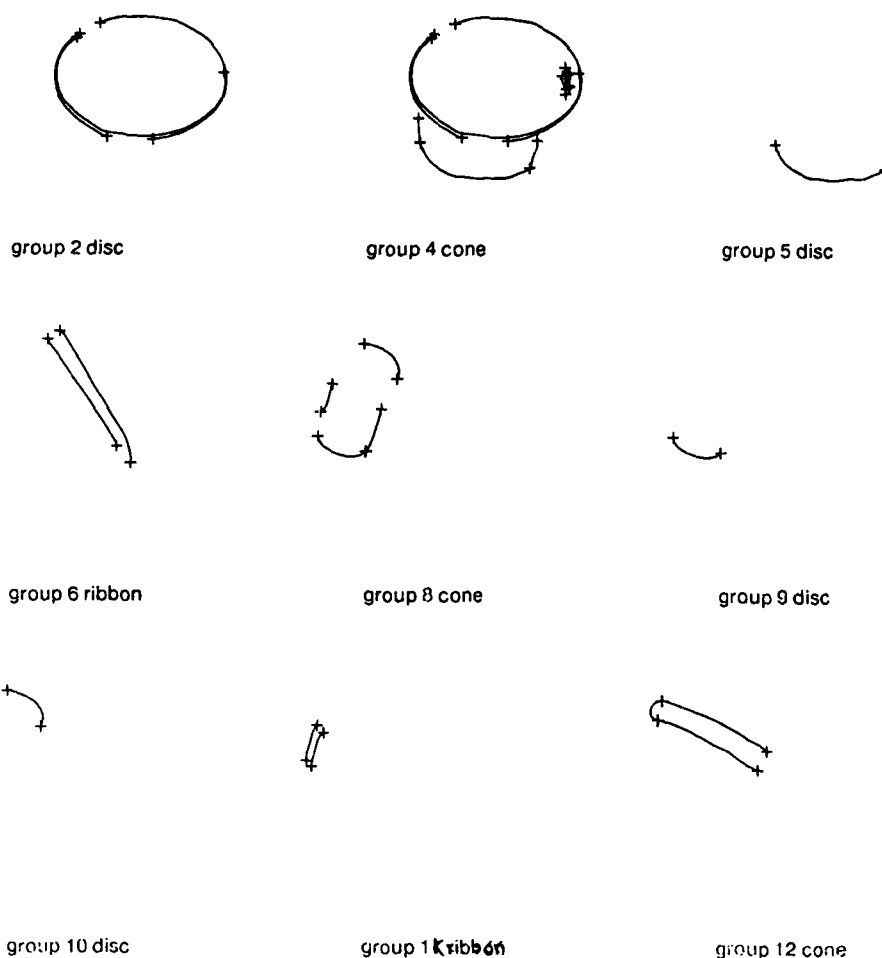


Figure 7: The result of the contour grouping for Figure 5 a). There are gaps in the numbering because one group may be subsumed in a later hypothesis, as the binary relations involving different contours are examined. Group 2 is formed by a set of cocircular contours, and suggests a (nonexistent) disc. Group 4 suggests the frustum of a cone. It was formed because the bottom circular contour was coaxial with the top ones, while the straight sides extended between top and bottom. Groups 8 and 12 were aggregated in the same way, although group 12 lacks a bottom. Groups 5, 9, and 13 are discs suggested by lone circular contours. Groups 6 and 11 are ribbons. The ribbon classification avoids too-early commitment to shape, since there are no cross-contours to indicate whether the surface is curved. Group 6 lacks the cross-contour because the contour at the end was discarded as unreliable because it was too short.

To solve the transform match in a general manner is not a straightforward calculation of coordinate transformation matrices for three reasons. First, when the component surfaces include some symmetry, such as in cylinders and cones, then the placing the coordinate frame is not unique. Therefore, the computed transforms may appear different

even for the same geometrical situation. Secondly, the objects may have parts connected by linear or rotational articulations, such as scissors. We need a method of representing, calculating, and comparing transforms which accommodates objects with articulations. Finally, due to the measurement errors, the calculated transforms won't be

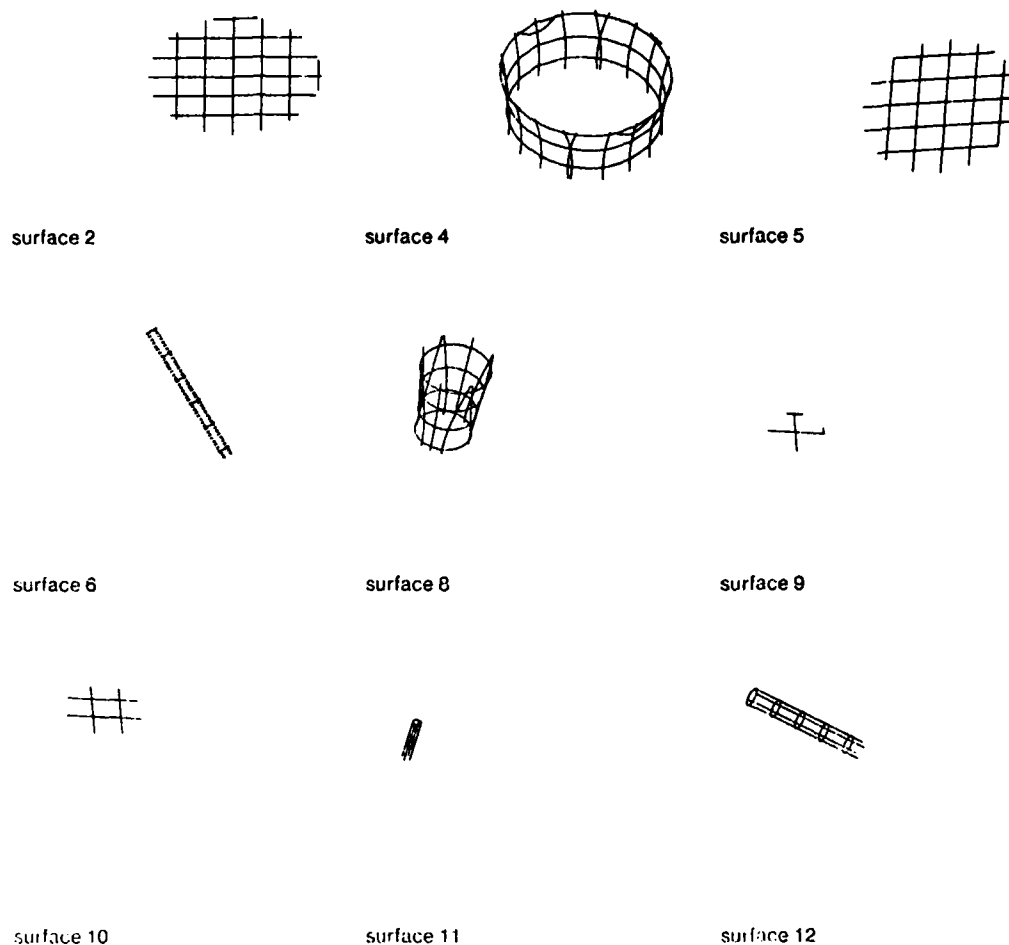


Figure 8: Surfaces proposed by the contour groups in Figure 7. Surfaces 4 and 8 are internally represented as rings cut from quadric surfaces, but are symbolically classed as (frustrums of) cones. Surfaces 6 and 11, derived from ribbon contour groups, are seen to be cylinders.

exactly the same even when they should be. Thus a method is required which can tell whether the transform is approximately the same. This is especially important for the cases of objects including articulations, because element-by-element comparison does not work. Smith [12] and Tomita and Kanade [13] discuss these three problems in more detail, and present partial solutions. A technique of merging descriptions from image sequences has been developed also in [5] for the domain of aerial photo interpretation.

7. Summary

This paper has described a method for 3D range-data analysis, which uses coherencies among contours, surfaces, and scenes to generate object descriptions. Specifically, the following points have been discussed:

- Taxonomy of contours in light-stripe images. This helps us to understand what causes each contour; whether it occludes or is occluded. The detection can be done with the initial deflection image, prior to conversion to three dimensions.

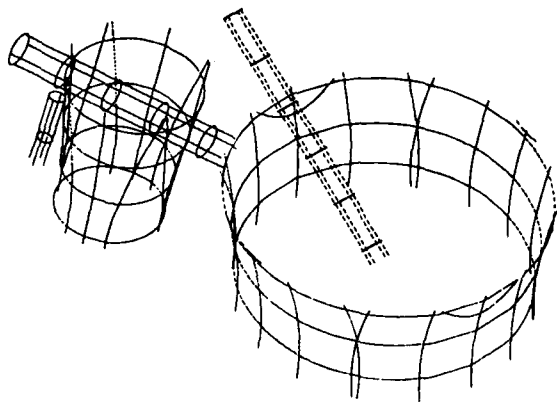


Figure 9: Accepted surfaces from Figure 8

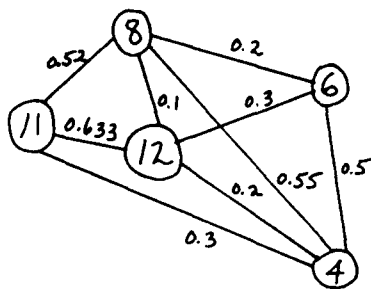
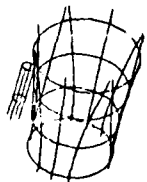
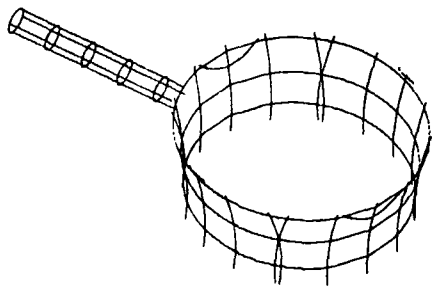


Figure 10: Relationship graph for surfaces

surface 4,12



surface 8,11



surface 6

Figure 11: Objects in the scene: surface groupings

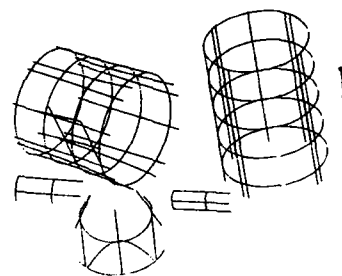
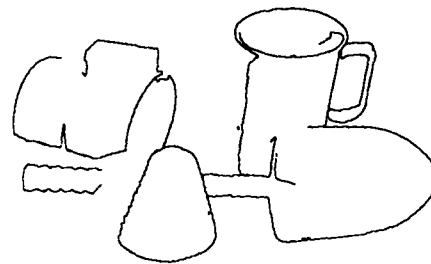
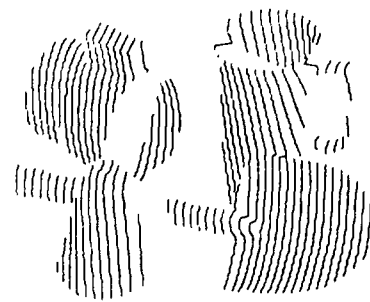


Figure 12: Results of analysis for another scene

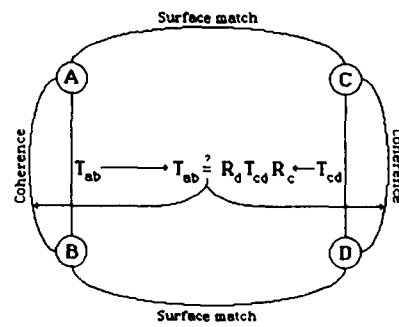


Figure 13: Matching surface pairs from different images

- Explicit use of the hierarchy of contour, surface, object and scene to analyze range-data imagery.
- Development of a method for utilizing coherent relations among lower-level elements as shape cues to aid extraction of higher-level elements.
- Data-driven autonomous generation of object descriptions without specific prestored models.

We plan to further develop and implement a systematic method of combining images of different scenes to obtain consistent descriptions of objects.

7.0.1. Acknowledgements

The light-stripe rangefinder images used in this research were provided by Dr. T. Oshima, Electrotechnical Laboratory, Japan. We thank Martial Hebert, Martin Herman, and Fumiaki Tomita for useful discussions.

References

- [1] Gerald J. Agin, Thomas O. Binford.
Computer description of curved objects.
In Proc. 3rd Int. Joint Conf. Artificial Intelligence,
pages 629-640. , 1973.
- [2] Binford, T. O.
Survey of Model-Based Image Analysis Systems.
Robotics Research 1(1):18-64, Spring., 1982.
- [3] Robert C. Bolles, Patrice Horaud, Marsha Jo Hannah.
3DPO: A Three-Dimensional Part Orientation
System.
In Proc. 8th Int. Joint Conf. Artificial Intelligence,
Karlsruhe, W. Germany, August, 1983.
- [4] O.D. Faugeras, M. Hebert.
A 3-D Recognition and Positioning Algorithm Using
Geometrical Matching Between Primitive
Surfaces.
In Proc. 8th Int. Joint Conf. Artificial Intelligence,
Karlsruhe, 1983.
- [5] Martin Herman, Takeo Kanade, Shigeru Kuroe.
Incremental Acquisition of a Three-Dimensional
Scene Model from Images.
IEEE Trans. Pat. Anal. Mach. Intell. PAMI-6(3),
May, 1984.
- [6] Kanade, T.
Recovery of the Three-Dimensional Shape of an
Object from a Single View.
AI 17:409-460, 1981.
- [7] Lowe, D. G. and Binford, T. O.
The Perceptual Organization of Visual Images:
Segmentation as a Basis for Recognition.
In DARPA Image Understanding Workshop, pages
203-209. 1983.
- [8] R. Nevatia, T. O. Binford.
Structured descriptions of complex objects.
In Proc. 3rd Int. Joint Conf. Artificial Intelligence,
pages 641-647. Stanford, Calif., 1973.
- [9] R. J. Popplestone, C. M. Brown, A. P. Ambler, G. F.
Crawford.
Forming models of plane-and-cylinder faceted
bodies from light stripes.
In Proc. 4th Int. Joint Conf. Artificial Intelligence,
Tbilisi, Georgia, USSR, 1975.
- [10] Steven A. Shafer.
*Shadow Geometry and Occluding Contours of
Generalized Cylinders*.
PhD thesis, Carnegie-Mellon University, May, 1983.
- [11] Yoshiaki Shirai, Motoi Suwa.
Recognition of polyhedrons with a range finder.
In Proc. 2nd Int. Joint Conf. Artificial Intelligence,
pages 80-87. 1971.
- [12] Smith, D. R.
Autonomous Scene Description with Range Imagery.
PhD thesis, Carnegie-Mellon University,
1984(toappear).
- [13] Tomita, F. and Kanade, T.
A 3D Vision System: Generating and Matching
Shape Descriptions in Range Images .
*In Second International Symposium on Robotics
Research*. 1984 (to appear).

DESCRIPTION OF 3-D SURFACES USING CURVATURE PROPERTIES¹

G. Medioni and R. Nevatia

Intelligent Systems Group
Department of Electrical Engineering
University of Southern California
Los Angeles, California 90089-0272

1. ABSTRACT

We describe an approach for describing 3-D surfaces by using the surface curvature. Surface curvature completely determines a surface. Further, we suggest that simple properties of curvature, such as points and lines where principal curvature is a maximum, correspond to important physical properties of the surface. Several examples are given, however, the process of interpretation of curvature properties has not been fully completed.

2. INTRODUCTION

We are interested in the description of 3-D surfaces and objects, assuming that range data (i.e. the 3-D positions) of the points on the visible surface are available, by say the use of a laser range finder. We also assume that this data is "dense", in the sense of being sampled on a certain grid and not just at discontinuities (as may be the case for uninterpolated stereo edge data); interpretation from sparse data is discussed in another paper from our group in these proceedings [1].

To generate useful descriptions, we need a useful representation. In general, such a description should be suitable for the task of object recognition and position identification. It should be rich, so that similar objects can be identified, stable, so that local changes do not radically alter the descriptions, and have local support so that partially visible objects can be identified. It should also enable us to recreate, from its features, a shape reasonably close to the original one.

Generalized cones have come to be recognized as an important class of representations, that satisfy the above requirements, particularly for complex objects, which are described as assemblies of smaller objects [2]. However, generalized cones are "volume" descriptions and may not be suited for objects that are essentially surfaces, such as a metal sheet, or for relative smooth, "featureless" surfaces such as a turbine blade. In this paper, our interest is in inspection of such high precision surfaces (our representation may also turn out to be an important step in generating generalized cone descriptions).

Our approach to describing surface is to consider the curvature of the surface described by the range image, and more specifically to identify and extract significant

changes of this curvature. It is therefore related to the "Curvature Primal Sketch" of Asada and Brady [3] describing closed planar curves, but we are now describing surfaces in 3-D. Our approach allows for description of local properties, in contrast to global methods such as "Extended Gaussian Images" [4].

3. CURVATURE REPRESENTATION

3.1. Mathematical Background

A curve in E^3 (3 dimensional Euclidian space) is uniquely determined by 2 local quantities called curvature and torsion. Similarly, a surface in E^3 is determined by 2 local invariant quantities called the first and second fundamental forms [5].

Let $z=f(x,y)$ be a coordinate patch on a surface of class ≥ 1 (f is in C^1 and the rank of the Jacobian matrix is 2). By convention, let $dz=f_x dx + f_y dy$, with $f_x=\partial z/\partial x$, $f_y=\partial z/\partial y$

dz has the property that

$$f(x+dx, y+dy) = z + dz + o((dx^2+dy^2)^{1/2}) \quad (1)$$

So, dz is a 1st order approximation of $f(x+dx, y+dy)-z$

Consider the quantity

$$I=dz \cdot dz = (f_x dx + f_y dy) \cdot (f_x dx + f_y dy)$$

$$= (f_x \cdot f_x) dx^2 + 2(f_x \cdot f_y) dx dy + (f_y \cdot f_y) dy^2$$

$$= E dx^2 + 2F dx dy + G dy^2 \quad (2)$$

$$\text{where } E=f_x \cdot f_x \quad F=f_x \cdot f_y \quad G=f_y \cdot f_y \quad (3)$$

I is known as the first fundamental form.

Let us now suppose that $z=f(x,y)$ is a patch of class ≥ 2 . We can define a unit normal at each point $N=f_x \times f_y / |f_x \times f_y|$, then, $dN=N_x dx + N_y dy$. Consider the quantity

$$II = -dx \cdot dN = -(f_x dx + f_y dy) \cdot (N_x dx + N_y dy)$$

$$= -f_x \cdot N_x dx^2 - (f_x \cdot N_y + f_y \cdot N_x) dx dy - f_y \cdot N_y dy^2$$

¹This research was supported by Defense Advanced Research Projects Agency under contract numbers F33615-82-K-1786 and F33615-84-K-1404, monitored by Wright Patterson Aeronautical Labs

$$= Ldx^2 + 2M dx dy + Ndy^2 \quad (4)$$

$$\left. \begin{aligned} \text{where } L &= f_{xx} \cdot N_x = f_{xx} \cdot N \\ M &= -1/2 (f_{xy} \cdot N_x + f_{xy} \cdot N_y) = f_{xy} \cdot N \\ N &= f_{yy} \cdot N_y = f_{yy} \cdot N \end{aligned} \right\} \quad (5)$$

It is known as the second fundamental form, and $\delta = 1/2II$ is called the **osculating paraboloid** at each point P and its nature qualitatively describes the nature of the surface around P based upon the discriminant $LN-M^2$.

- **Elliptic case:** $LN-M^2 > 0$. Here, δ as a function of (dx, dy) is a elliptic paraboloid as shown in figure 1(a). The surface lies on one side of the tangent plane.
- **Hyperbolic case:** $LN-M^2 < 0$. δ is a function of (dx, dy) is a hyperbolic paraboloid as shown in figure 1(b). There exist 2 lines in the tangent plane through P which divide the tangent plane into 4 sections in which δ is alternately positive and negative.
- **Parabolic case:** $LN-M^2 = 0$ and $L^2+N^2+M^2 \neq 0$. δ as a function of (dx, dy) is a parabolic cylinder, as shown in figure 1(c). There is a single line in the tangent plane through P along which $\delta=0$, otherwise δ keeps a constant sign.
- **Planar case:** $L=M=N=0$. Here $\delta=0$ for all (dx, dy) .

At each point, surface shape can be completely described by the six coefficients E, F, G, L, M, N mentioned above. These coefficients, however, are not independent, and it seems better to use more meaningful descriptors, such as the **principal curvatures**.

Let us draw a curve C of class C^2 onto our surface, passing through point P. The normal curvature k_n of C at P is the projection of the curvature vector k of C at P on N. $k_n = k \cdot N$.

It is easy to prove that $k_n = II/I$.

The two directions for which the values of k_n take on maximum and minimum values are called the **principal directions**, and the corresponding normal curvatures, κ_1 and κ_2 , are called the **principal curvatures**.

The principal directions can be shown to be orthogonal, and that a number k is a principal curvature if and only if k is a solution of the equation

$$(EG-F^2)k^2 - (EN+GL-2FM)k + (LN-M^2) = 0 \quad (6)$$

A direction (du, dv) is a principal direction at P if and only if du and dv satisfy

$$(EM-LF)du^2 + (EN-LG)dudv + (FN-MG)dv^2 = 0 \quad (7)$$

We also need to define the Gaussian curvature K at P

$$K = \kappa_1 \kappa_2 = \frac{LN-M^2}{EG-F^2} \quad (8)$$

We can also prove that $EG-F^2 > 0$, therefore the sign of K agrees with the sign of $LN-M^2$. Thus, a point on a surface is **elliptic** if and only if $K > 0$; **hyperbolic** if and only if $K < 0$; **parabolic** or **planar** if and only if $K = 0$.

3.2. Choice of Representation

For shape description, it is not sufficient that the representation specify the original shape completely. We propose to use certain points and lines that have distinguished and invariant properties, and the process of description requires making these points and lines explicit. They can be useful for matching (registration) or as the key steps in computing higher level descriptions, such as generalized cones. We suggest that the distinguished points are essentially the points of extremal curvature, or those of curvature changes, and the lines connecting such contiguous points. We investigate the properties of such points and lines for specific configurations in surfaces below.

For the curvature at a point, we will use the principal curvatures (κ_1 and κ_2) and the Gaussian curvature ($\kappa_1 \kappa_2$). The principal curvatures have a magnitude as well as direction, with the two principal directions being orthogonal.

4. COMPUTING CURVATURE

First, a few comments on computing curvature. Since we have discrete data, we must compute differences rather than derivatives. We compute the first differences in the x and y directions by convolving the range images with masks as shown in Fig. 2(a). The second differences in x, y, and the cross-derivative $\partial^2 f / \partial x \partial y$ are computed by convolving with masks shown in Fig. 2(b). These masks are obtained by differentiating Lagrange's polynomials. The principal curvatures and their orientations are obtained by solving equations (6) and (7).

Computation of curvature is likely to be highly noise sensitive. To decrease the effects of noise, we can use a large support for computing differences, at a possible cost in accuracy of localization. We convolve the image with rotationally symmetric Gaussian masks of different variance, σ . The different sizes of the filter give us curvature at different scales, and are in fact helpful in interpreting the results, (as in [3] and [6] for other applications). The width of the mask is chosen such that the volume under the truncated Gaussian is very close to 1 (6σ is a good approximation).

For example, if $\sigma=1$, the 5x5 filter has a volume of 0.982 and coefficients in the upper left corner shown in figure 3.

5. INTERPRETING CURVATURE

5.1. Introduction

We are interested in three major properties of surfaces:

- **Range Discontinuities** (or "jump" boundaries): these are typically the occluding contours of a surface
- **Surface Discontinuities**: these correspond to folds or cuts in a surface, such as the edges of a polyhedral object
- **Points of curvature maxima**: these are physical invariants of a surface

These are the properties of the surfaces that we wish to identify, we now need to verify that they translate easily into curvature properties, which is what we measure.

The first, and simplest attribute of curvature properties is the sign of the Gaussian curvature. There have been empirical observations that the shape of the regions of constant Gaussian curvature sign roughly reflect the overall shape of an object [7]. We will therefore detect zero-crossings of the Gaussian curvature, together with their angle. This feature allows us to reconstruct exactly the region of constant Gaussian curvature sign.

Gaussian curvature, however, captures only a small part of the total information, and is totally inappropriate to describe the large subclass of cylindrical objects. Indeed, the Gaussian curvature is 0 for such objects, regardless of the cross-section shape. Such objects are then described by the variations of the maximum principal curvature, and more succinctly by the location of the zero-crossings and maxima of this function.

These are the features we will be using for shape description: zero-crossings of the Gaussian curvature, zero-crossings and maxima of the largest principal curvature.

In the introduction, we suggested that objects are naturally segmented at the points of maximum curvature, with no mention of sign changes. The explanation is simple: the curvature we mentioned previously is the one we perceive by rolling a finger on the object, for example, and not the one we compute using digital filters. In the case of a jump boundary, for instance, we will see that, due to the digital computations, the discontinuity is located by the zero-crossing position between a large positive and a large negative response.

Let us now generate some instances of interesting surface properties and observe the corresponding curvature transitions.

5.2. Some Special Cases

5.2.1. Step edge

An example of step edge is shown in figure 4. We can consider it as a 2D problem, the minimum curvature being 0 almost everywhere (planar patch). In the continuous case, the first derivative should be 0 everywhere, except at the discontinuity where we have a delta function. The second derivative is also 0 everywhere, except at the discontinuity where we have 2 delta functions of opposite sign. In the discrete case, the maximum curvature is characterized by 2 maxima of opposite sign and a zero-crossing at the discontinuity location. As σ increases, the value of these maxima decreases and they move away from the (fixed) zero crossing locations as shown in figure 5.

5.2.2. Surface discontinuities

This corresponds to a fold in a surface. We can identify different types of such an instance, depending on

the sign of the curvature on each side of the fold, as illustrated in figure 6. The common characteristics are a maximum of the curvature near the location of the fold, decreasing with larger values of σ . Zero-crossings may appear, depending on the curvature sign, as shown in figure 7, representing the variation of κ_1 for the fold shown in figure 6(c). If the curvature is 0 on either side, we have a polyhedral edge, and the corresponding behavior of κ_1 is shown in figure 8 for different values of σ .

5.2.3. Maximum of curvature

This is typically illustrated by an elliptical cylinder, where the maximum curvature keeps a constant sign and goes through a very smooth maximum.

5.2.4. Others

We could define more types of transitions, as in [3], some of them ambiguous at certain scales; it is, however, more interesting to concentrate on the problems arising when the Gaussian curvature is non-zero.

The case of positive Gaussian curvature is rather straightforward, and only maxima of κ_1 can appear in these zones. False maxima may appear in the neighborhood of umbilical points (for which $\kappa_1 = \kappa_2$), when κ_1 and κ_2 exchange their role, undergoing a sudden 90° change in orientation.

In negative Gaussian areas, the situation is more complex: Around points where the magnitude of κ_1 and κ_2 is the same, we can have reversals of (κ_1, κ_2) , that is the larger curvature becomes κ_2 , undergoing a 90° change and creating a zero-crossing in the curves representing the maximum and the minimum principal curvatures, even though the surface itself is smoothly changing. This effect is shown in the "vase" example.

6. RESULTS

Our implementation so far consists of the computation of the principal curvatures, the extraction of zero crossings from the Gaussian curvature and the maximum curvature images, and the localization of the maxima in the maximum curvature image. We have chosen two representative examples to illustrate the strong link between curvature behavior and the features we wish to detect:

The first one is the range image of a straight cylinder with an hexagonal base. It is a good example of zero Gaussian curvature surface, showing polyhedral edges and jump boundaries.

The second one is the range image of a vase, which can be thought of as a Straight Homogeneous Circular Generalized Cylinder [8], meaning that it has a circular cross section, and a straight axis. It presents zones of positive and negative Gaussian curvature, jump boundaries, surface discontinuities and zero-crossings due to reversals of (κ_1, κ_2) .

Figures 9 and 10 present the processing performed on each of these range images, and are organized as follows:

- (a) is a graphic presentation of the object against the background
- (b) is a "needle map" representation of the surface orientation, obtained from the first order differences.

- (c) is a representation of κ_1 , the maximum principal curvature. The length of the needle is proportional to the magnitude of the curvature. The curvature displayed here is the one obtained without any smoothing of the data.
- (d-g) show the zero-crossings of κ_1 after smoothing the original data with a σ of 0, 0.5, 1 and 2 respectively.
- (h-k) show the maxima of κ_1 after smoothing the original data with a σ of 0, 0.5, 1 and 2 respectively.

Finally, figures 10(l-o) show the zero-crossings of the Gaussian curvature for the "vase" image. For the hexagon image, the Gaussian curvature is 0 everywhere.

We can make the following comments on these figures:

- The horizontal lines in figures 9(d) through 9(g) detect zero-crossings of κ_1 , which varies in these areas between -10^{-7} and $+10^{-7}$, so that these lines should be ignored.
- In this first example, we clearly see that the maxima of κ_1 identify the polyhedral edges accurately for all values of σ . Also, the jump boundary corresponds to a zero-crossing of κ_1 flanked by two maxima of opposite sign, receding from the zero-crossing as σ increases, as predicted.
- On the vase example, we see that the computation of curvature is very sensitive to local distortions, such as quantization effects visible in 10(a) and reflected in the processed views 10(d), 10(h) and 10(l).
- The Gaussian curvature is a good qualitative description of the object as shown in 10(l-o).
- It is the zero-crossings of κ_1 (and not of the Gaussian curvature) which accurately correspond to the non-jump boundary of the object, and the accuracy decreases as σ increases.
- In the bottleneck area of the vase, we detect zero-crossings of κ_1 , even though no significant changes of the surface occur. They are detected as a consequence of the "swap" between κ_1 and κ_2 , since the two principal curvatures have opposite sign, and one accompanied by a 90° rotation of κ_1 and κ_2 . This phenomenon is well illustrated by figure 11, showing a profile of κ_1 and κ_2 along column 39.

7. FUTURE RESEARCH

Obviously, the material described in this paper only represents the groundwork for a complete feature extraction task, and serves as a validation of our ideas regarding curvature as a powerful, local support primitive.

What needs to be done is

- The interpretation of significant zero-crossings and maxima as jump boundaries, edges or simply extrema. For instance, the zero-crossing of κ_1 associated with the jump boundary of the vase example is not accompanied by a 90° angle change, except for $\sigma=2$.
- The integration of the detected primitives using various values of σ . This step will also be useful to

perform the interpretation: the maxima for a jump boundary move away as σ increase.

- The linking of identified primitives into connected components.
- The reconstruction, from our features, of an object which should be reasonably close to the original.
- The abstraction of such features, leading to a concise, natural description of the object, such as a generalized cone.

References

1. Nevatia, R., "Image Understanding Research at USC: 1983-84," *Proceedings of Image Understanding Workshop*, 1984.
2. Binford, T.O., "Visual Perception by Computer," *IEEE Conference on Systems and Controls*, December 1971.
3. Asada, H. and Brady, M., "The Curvature Primal Sketch," *Proceedings of the 2nd IEEE Workshop on Computer Vision: Representation and Control*, Annapolis, MD, May 1984, pp. 8-17.
4. Horn, B.K.P., "Extended Gaussian Images," Tech. report A.I. Memo No. 740, Massachusetts Institute of Technology, July 1983.
5. Lipschutz, M., *Differential Geometry*, McGraw-Hill, 1969.
6. Witkin, A.P., "Scale-Space Filtering," *Proceedings of Seventh IJCAI*, Karlsruhe, West Germany, August 1983, pp. 1019-1022.
7. Terzopoulos, D., *Multiresolution Computation of Visible-Surface Representations*, PhD dissertation, Massachusetts Institute of Technology, Departments of Computer Science and Electrical Engineering, January 1984.
8. Shafer, S.A., "Shadow Geometry and Occluding Contours of Generalized Cylinders," Tech. report, CMU Report CS-83-131, May 1983.

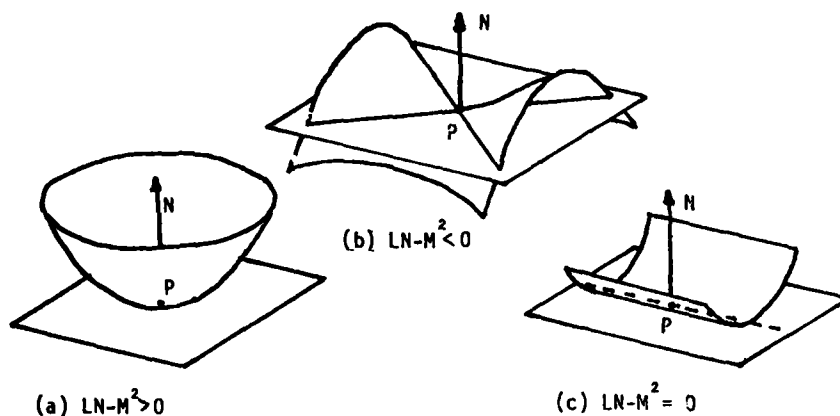


Figure 1: Different types of surface

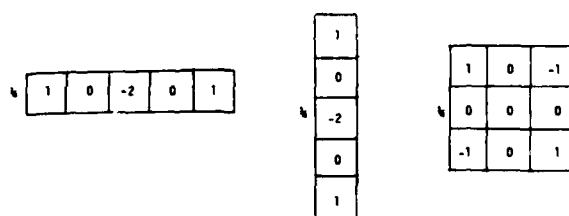
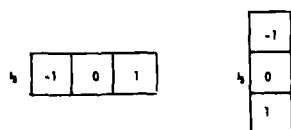


Figure 2: Masks to compute 1st and 2nd order differences

Upper left corner of Filter:

.2915024e-2	.1306423e-1	.2153928e-1
.1306423e-1	.5854983e-1	.9653235e-1
.2153928e-1	.9653235e-1	.1591549

Sigma: 1.000000 Scale: 1.000000
Size:5 Volume: .9818148

Figure 3: Gaussian filter

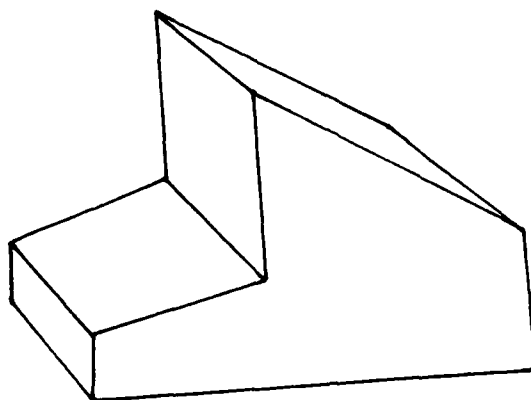


Figure 4: A step edge

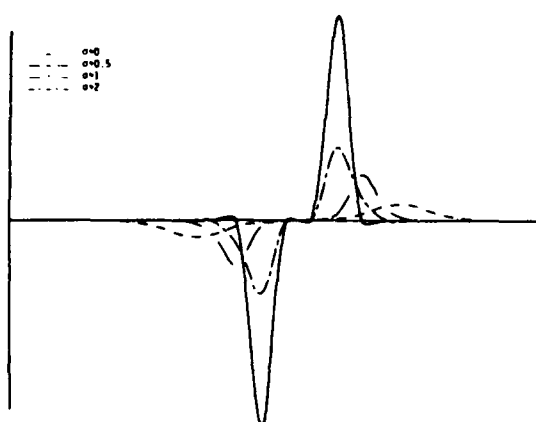


Figure 5: variation of κ_1 for a step edge

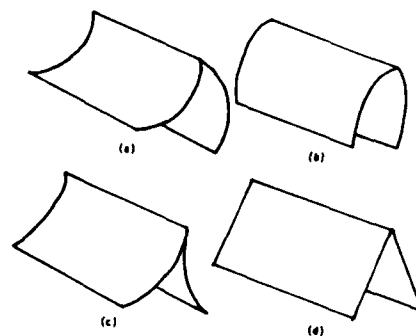


Figure 6: Different types of surface discontinuities

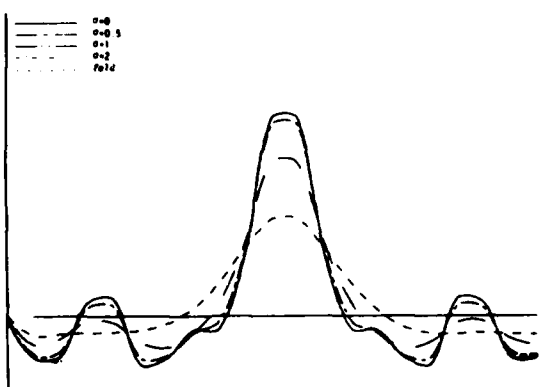


Figure 7: variations of κ_1 for one type of fold

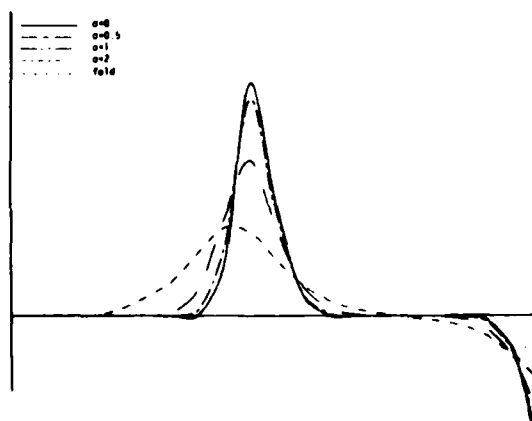


Figure 8: variations of κ_1 for a polyhedral edge

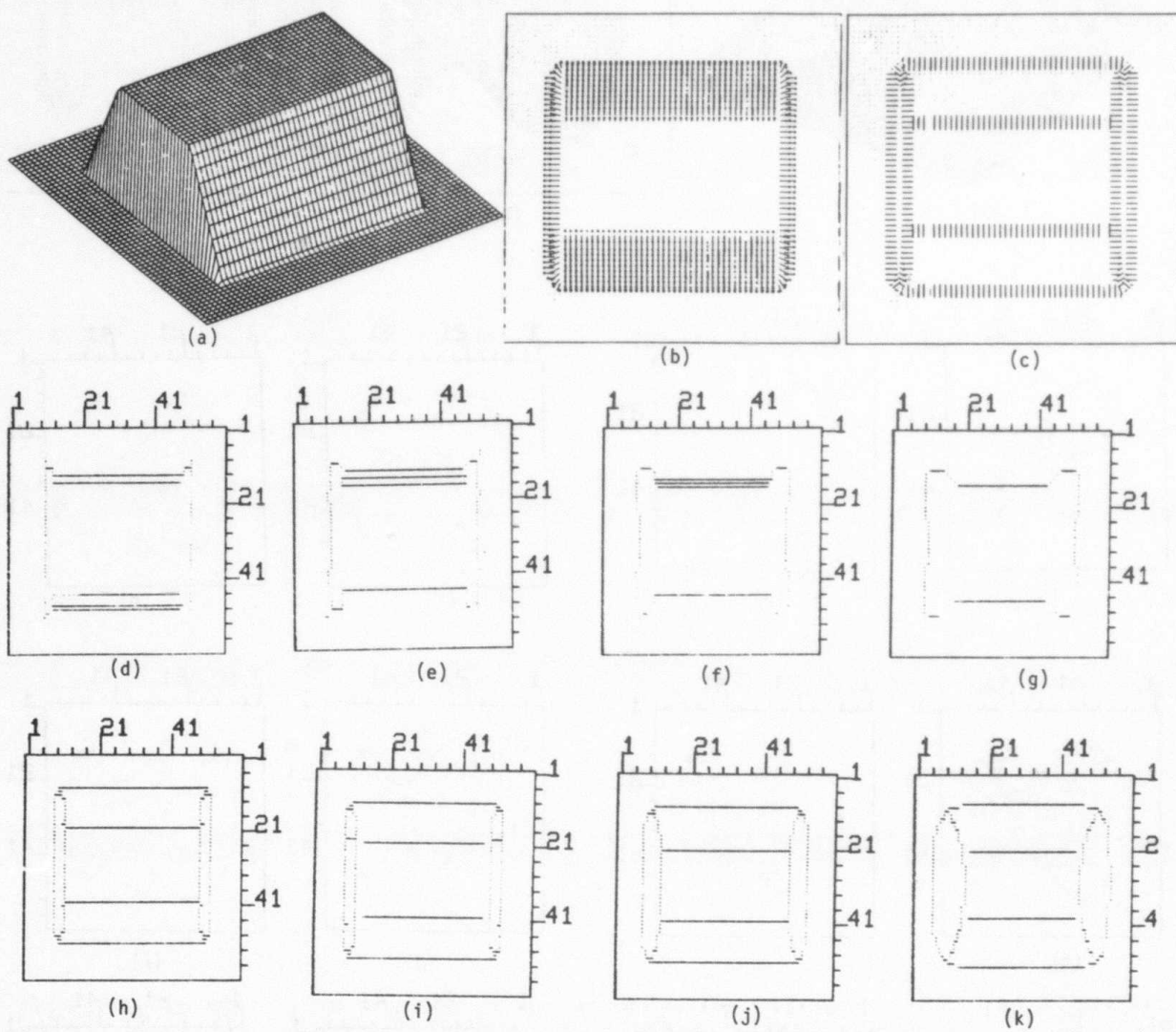
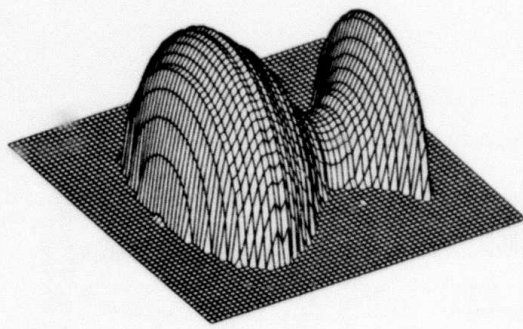
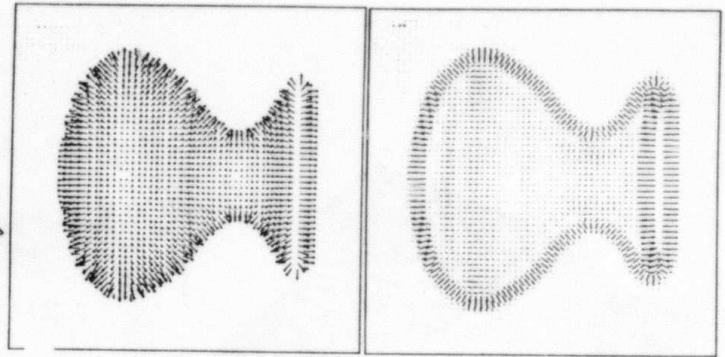


Figure 9: Hexagon Example

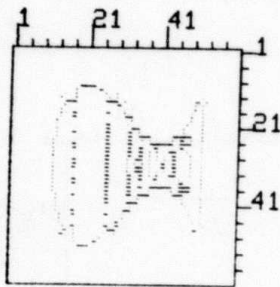


(a)

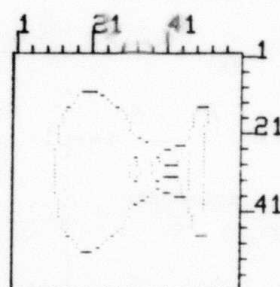


(b)

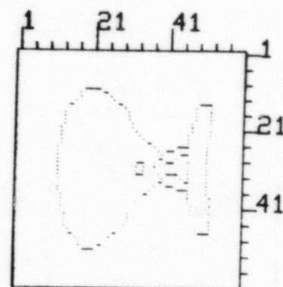
(c)



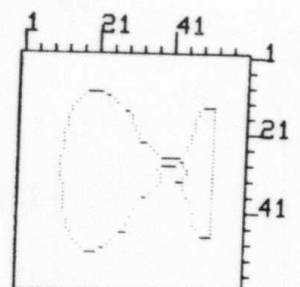
(d)



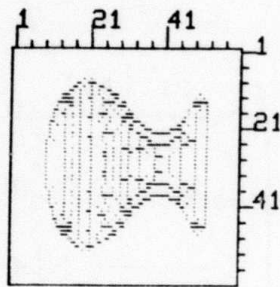
(e)



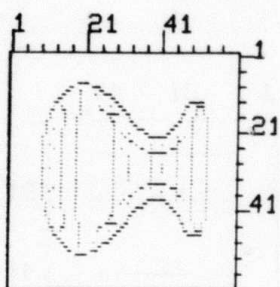
(f)



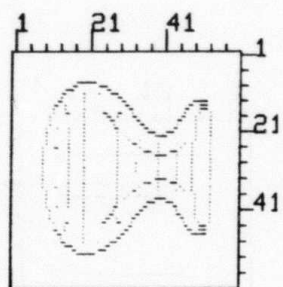
(g)



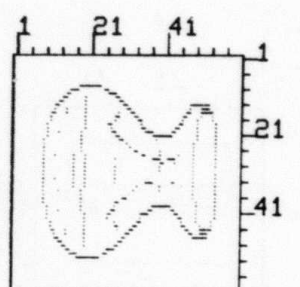
(h)



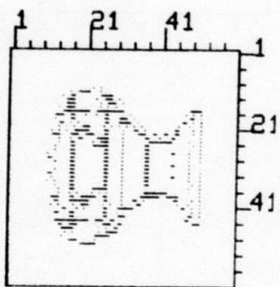
(i)



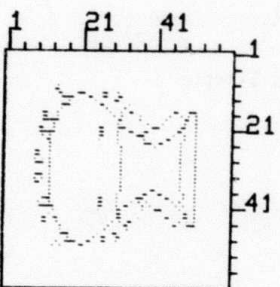
(j)



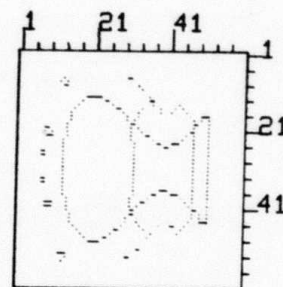
(k)



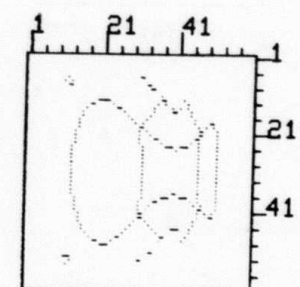
(l)



(m)



(n)



(o)

Figure 10: Vase Example

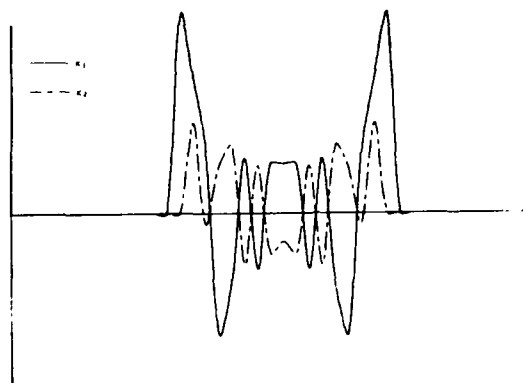


Figure 11: Zero-crossings of κ_1 and κ_2 in smooth negative Gaussian Curvature zone

**The ASTERIX-System:
A FEATURE BASED APPROACH TO THE CORRESPONDENCE PROBLEM**

Leonie S. Dreschler-Fischer and Ernst E. Triendl

Stanford Artificial Intelligence Laboratory
Stanford University, Stanford Ca. 94305

ABSTRACT

A new feature based approach to the correspondence problem is presented. The system described is a general system which can be used both for motion-sequences and stereo-pairs. It is shown how by grouping and grading of features the matching process can be guided by a plan to deal with ambiguous feature constellations in an "intelligent" way.

INTRODUCTION: A VISION SYSTEM FOR A MOBILE ROBOT

The ASTERIX-System is planned to be part of the vision system for a mobile robot. The robot will be equipped with a pair of stereo-cameras, acoustic sensors and a laser range finder. Hence the vision system has to handle stereo pairs as well as motion sequences and should be able to incorporate depth clues from other sensors.

Both motion and stereo vision have the correspondence problem in common. The correspondence problem in both domains differs only in the type of the constraints, which can be applied to solve the problem. The main task however, selecting features in all frames of a motion sequence or a stereo pair, is the same. That is why the ASTERIX-system is based on a general correspondence algorithm which can be applied to motion sequences as well as to stereo pairs.

Specialized knowledge about disparity constraints is kept in separate program modules (constraint sources), which evaluate e.g. epipolar lines (stereo) [1, 6] or velocity predictions (motion)[9]. These separate constraint sources provide also an easy way to feed constraints derived from other kinds of sensors into the system. Thus a change in the robot configuration affects only the constraint sources and the rules how to apply them, but not the general concept of the system.

Another advantage of this system structure is that in the case of motion sequences the constraint sources can change according to the knowledge acquired so far. In the first frames of an image sequence only very general heuristic constraints can be used, but as more and more about the scene is known more sophisticated the constraints can be applied.

The matching is based on local features like corners and dots [5], L-, Y-, X- and T-junctions [11, 12] for the following reasons:

- The extraction of these local features can be done very fast, since most of the required computation time is used for the convolution of the image with constant operators [8], which can be done with an array processor nearly in real-time.
- Since the purpose of the matching is to compute the 3D-coordinates of the physical object corresponding to the matched features, it is very convenient to match features which have unique coordinates, like corners or dots.
- On the other hand should the features to be matched provide enough information to reduce search time and ambiguities (compare with approaches which rely entirely on grayvalue correlation [6, 7, 8]).

- Small local features are less likely to be distorted by occlusion or perspective distortions than features that extend over large areas of the images. Hence they are easier to match than higher level features like polygons, e.g. if from a square only three corners are visible you cannot match a square but you still can match three corners of the square.
- The classification of the corners allows to exclude T-junctions from the stereo analysis, which might be caused by occlusion and must not be used for depth measurements.

The matching strategy in ASTERIX is the most important part of the system. ASTERIX first inspects the feature constellations in both frames and plans a matching sequence which seems to be the most promising for the situation.

Consider for example the image of a checker board: All corners in the inner part of the board have look-alike neighbors. Any matching strategy based on optimizing a similarity measure within a local neighborhood are bound to come up with ambiguous matches or worse, will force an arbitrary decision and select the best match according to the similarity measure, which might depend purely on the noise in the image. Matching strategies based on a global optimization criterion [2, 13] on the other hand have the tendency in case of ambiguities to force occasionally some wrong matches for the sake of a globally better optimization. Ullman's "minimal mapping" will get the matches for the checker board totally wrong if the disparity is similar to the size of the squares.

ASTERIX uses a different strategy: The system first scans both frames independently and groups the features into classes of similar features, which might cause ambiguities. In the case of the checker board ASTERIX would see immediately that there are lots of look-alike corners in the inner parts of the board, but that the four corners of the board itself are unique. So the system starts matching by matching one of these unique corners first. Then the neighboring corners are matched by following the edges that connect them, until all parts of the image are matched or no more constraints can be derived. Ambiguities which cannot be solved at this stage of analysis are forwarded to the higher levels of analysis, rather than forcing a decision based on low level clues only.

The way ambiguities are solved in ASTERIX is related to graph matching techniques like the tracking algorithm used by Radig and Zach [10]. Both strategies are feature based, evaluate all possible ambiguities and derive constraints from structural knowledge like neighborhood relations. The difference is that graph matching strategies compare features *between* the frames to be matched whereas ASTERIX groups the features and compares them mainly *within* the frames. Graph matching is very mechanical whereas the grouping generates a plan for a data-driven matching sequence.

The planning of the matching sequence in ASTERIX can be compared to the way a child tries to solve a jigsaw puzzle. Usually the child will start with pieces that catch the eye because they are strikingly different from all other pieces in either shape or color, and the more advanced child will organize his pieces and sort them according to color and shape.

FEATURE EXTRACTION AND GROUPING

Feature extraction is done in three steps: First an interest operator [5] and an edge detector [12] are applied to the images and basic features (edgels and points of interest: dots, corners) extracted. The interest operator labels the points of interest as dark objects on a light background or vice versa and computes also the orientation of the corners and a measure of the quality. The edgels are labeled with the average grayvalues on both sides of the edge, the orientation, and also a quality measure. All these descriptions are used to match features.

In a second step edgels are tracked to find edges, their endpoints and intersections. Neighboring endpoints, intersections or points of interest are combined to new features and if possible labeled according to their junction type as (Y, L, T, X...) [11].

Finally, the features found in each frame are grouped into classes of features which are similar according to the similarity function which is used for matching. This grouping is done by computing the minimal spanning tree, so that the number of groups is variable.

The similarity function is based on the weighted average of the similarity of the basic features; to compute, for example, the similarity of two Y-junctions, the similarities of the three edges meeting at the junction to there corresponding edges are averaged.

GRADING AND MATCHING

Since the features are grouped into equivalence classes, matching takes place between classes of features rather than between features itself. The first step of matching is grading of the classes according to abundance and prominence of features. The matching sequence is determined by the grades of the classes. The classes with only few but very prominent features (high contrast) are matched first to get reliable and unique feature matches. From these initial matches new constraints are derived and propagated into the more ambiguous classes. The constraint propagation follows the edges which connect the corners and junctions.

The grouping and grading of features does not necessarily reduce the computation time for the matching. The comparison of features within the frames takes about the same time as the comparison of features between the frames, but the grouping process saves the results of the comparisons in a more useful way, so that a plan can be derived to solve ambiguous constellations.

IMPLEMENTATION AND RESULTS

The edge operator and the interest operator are implemented in C, all other parts of the system are running in SLISP. The hardware implementation of the convolution is in progress.

The feature extraction and the grouping and grading of features have been tested on real-world scenes (machine parts, telephone) with quite satisfactory results; the constraint system is still under development, and the matching has been tested so far only on artificial data, but further results are expected soon.

REFERENCES

1. H. Baker, Progress in Stereo Mapping, Proc. Image Understanding Workshop, April 1983.
2. S. T. Barnard and W. B. Thompson, Disparity Analysis of Images, *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-2 (1980), 333-340; see also Technical Report 79-1 (January 1979), Computer Science Department, University of Minnesota, Minneapolis, MN
3. P.R. Beaudet, Rotationally Invariant Image Operators, Proc. Int. Joint Conf. Pattern Recognition, Kyoto, Japan, November 7-10, 1978, pp. 579-583.
4. P. Blicher, Stereo Matching from the Topological Viewpoint, Proc. Image Understanding Workshop April 1983
5. L. Dreschler and H.-H. Nagel, Volumetric Model and 3D Trajectory of a Moving Car Derived from Monocular TV Frame Sequences of a Street Scene, *Computer Graphics and Image Processing* 20, 199-228 (1982)
6. D. B. Gennery, A Stereo Vision System for an Autonomous Vehicle, Proc. 5th International Joint Conference on Artificial Intelligence, MIT, Boston, Aug. 1977
7. M. J. Hannah, Computer Matching of Stereo Images, Ph.D. Thesis, Memo AIM 239 (July 1974), Stanford University, Stanford/Cal.
8. H.-P. Moravec, Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover, Ph.D. Thesis, Department of Computer Science, Stanford University, available as CMU-RI-TR-2 (September 1980) Robotics Institute, Carnegie-Mellon University, Pittsburgh, Pa.
9. H.-H. Nagel and W. Enkelmann, Iterative Estimation of Displacement Vector Fields from TV-Frame Sequences, Proc. Second European Signal Processing Conference EUSIPCO-83, Erlangen/FR Germany, September 12-16, 1983 (in press)
10. B. Radig, R. Kraasch and W. Zach, Matching Symbolic Descriptions for 3-D Reconstruction of Simple Moving Objects, ICPR-80, pp. 1081-1084
11. Ernst E. Triendl, The Edge Appearance Model in a Rule Based System, (to appear)
12. Ernst E. Triendl, Modellierung von Kanten bei unregelmässiger Rasterung, Proc. 1. DAGM-Symposium Oberpfaffenhofen/F.R.G. 1978, Springer Verlag Berlin Heidelberg New York 1978
13. S. Ullman, The Interpretation of Visual Motion, MIT Press, Cambridge/Mass. 1979

Acknowledgement:

This work is partially supported by the Air Force Office of Scientific Research under contract F49620-82-C-0092 and the Defense Advanced Research Projects Agency under contract N00039-84-C0211. It has also been partially sponsored by a grant of the German Academic Exchange Service (DAAD) and by the DFVLR. We thank Thomas Binford for many helpful discussions and suggestions and all other members of the robotics group who helped us to become familiar with the hardware and the software in the AI-Laboratory.

STEREO MODELING SYSTEM: A GEOMETRIC MODELING SYSTEM FOR MODELING OBJECT INSTANCE AND CLASS

Jun Takamura and Thomas O. Binford

Computer Science Department
Stanford University, Stanford, California 94305

Abstract

This paper describes an intelligent, user-friendly geometric modeling system which enables simple, fast building of 3D models for the Image Understanding system. First, an instance of an object is taught through the operations in which Generalized Cylinders are fitted to each part of an example of the object in 3D space formed by stereo images. Model descriptions of the instance are generated automatically after the operations. Second, class model of the object is learned inductively using the descriptions of the instances and model descriptions of the class are also generated. The generated model descriptions can be used as input of the ACRONYM system.

1. Introduction

As the research of model-based image understanding systems advances and these systems come to be applied to the recognition of various objects, an intelligent modeling system which can build 3D models efficiently is becoming necessary. Namely, a modeling system which enables simple, fast model building is required. Also, the system must be easy to learn in terms of how to build 3D models, and user-friendly from the viewpoint of the user-interface. A system for geometric modeling called Stereo Modeling System which satisfies these requirements has been designed and built.

In the ACRONYM system, 3D models were built through the text-based description language.¹ The problem was that model descriptions must be made by measuring various parameters, such as the size of a part of the object and the angle between parts, and by writing out long model descriptions. For example, to build a model of a relatively simple object like an aircraft, a couple of hundreds lines of model descriptions were required. It is not easy for a person who does not have enough knowledge of computers to handle the modeling language. Moreover, as the model descriptions are made by hand, errors in model descriptions are likely to occur. In order that a modeling system may be used widely and be applied to various objects, it is ideal that even a person who is not a computer expert can build 3D models of objects easily and without errors.

The 3D model has also been studied in the fields of CAD and Computer Graphics. However, the major difference between 3D models in these fields especially in CAD and in Image Understanding is that in the latter, objects to be modeled actually exist in the world. With this advantage, the system builds models of object instance and class from actual examples of the object through the following two successive stages.

- 1) A model of an object instance is taught interactively in 3D space formed by stereo images of an example and a stereoscope, i.e., geometric features of the object are specified directly in 3D space via the user-friendly interface, for instance voice and cursor, without symbolic descriptions, utilizing prior knowledge of the object.
- 2) The system learns the model of the object class inductively through the models of instances, and builds model descriptions of the object class automatically.

The model building of object instance is done by fitting Generalized Cylinders to each part of the example in 3D space, based on comparison of the displayed Generalized Cylinder and the actual ob-

ject. The teacher divides the object into several parts according to the required accuracy of approximation, specifies relations between parts, and builds the model of the whole parts efficiently, making use of the descriptions of the parts which are already defined. The system calculates parameters such as the size of the part or the angle between parts and generates model descriptions of the object instance. Section 2 describes the details of how to build instance models. Class models are built by generalizing model descriptions which are expressed in terms of algebraic constraints. Class models are also specialized by adding new constraints obtained through misrecognition experiences. The details will be described in Section 3. Specifying geometric features of objects directly in 3D space and teaching instance model using an example of the object is the most intuitive way to build 3D models. Moreover, making errors in model descriptions can be prevented by the procedure in which a building 3D model is displayed and always compared to the actual example. In Section 4, the user interface is described in detail through the actual example. The generated model descriptions can be used as direct object models of the ACRONYM system.

2. Fitting Generalized Cylinders in 3D Space

3D models are usually built by using standard components like cylinders, parallelepipeds, spheres, etc.. 3D models can be constructed to any degree of accuracy if the number of the faces is increased.² However, tractability and simplicity rather than accuracy of the model is important in image understanding for the processing stages of prediction and matching. Generalized Cylinders³ have been widely used as such powerful components for the 3D models of image understanding systems, for instance the ACRONYM. The Generalized Cylinder is also used as a primitive for building models in this system.

2.1. Generalized Cylinder Fitting Using Knots

The most intuitive way to build 3D models is to build models directly in 3D space viewing actual 3D examples. 3D input methods have also been studied in CAD and Computer Graphics.⁴ However, in the case of 3D models of image understanding, actual objects usually exist. In this system, 3D models are built efficiently in 3D space formed by stereo images, based on actual examples. Namely, a stereo display of an example is viewed through a stereoscope and the Generalized Cylinders are fitted to each part of the object by specifying knots in 3D space. The occurrence of errors in the model description can be prevented by the continuous comparison of displayed Generalized Cylinder with the actual object.

As the geometric modeling of synthetic objectives requires a high degree of accuracy, many knots must be specified in order to fit models to curves or surfaces. Although the surface can be fitted with any accuracy by polyhedral approximation, input and modification of the model is not easy. The fitting of curves and surfaces by Bézier or B-spline formulations also requires many control points. However, describing objects in terms of Generalized Cylinders for the purpose of image understanding purpose can be done simply by the following method.

- 1) From knots to a cross section

A Generalized Cylinder is defined by a cross section which sweeps along a spine, changing shape according to a sweeping rule as it continues to sweep. The types of Generalized Cylinders used in the

ACRONYM is shown in Table 1.

Cross-section	Sweeping rule	Spine
circle	constant	straight
square	linear	circular
rectangle	bilinear	non-perpendicular
regular-polygon		

Table 1. Types of Generalized Cylinder

A cross section is a plane and a plane can be determined by three knots as shown by an example in Figure 1. All the types of cross section in the ACRONYM can be fitted by three knots as illustrated in Figure 2.

2) From cross sections to a Generalized Cylinder

As described, a Generalized Cylinder is formed by a cross section's sweeping along a spine. All the types of Generalized Cylinder in the ACRONYM can be fitted by three cross sections at most. For example, three cross sections are required for the types of Generalized Cylinder which have a circular spine as shown in Figure 3. Two cross sections are required for the types of Generalized Cylinder which have a straight or non-perpendicular spine and have a sweeping rule other than the constant type. Consequently, nine knots are required to fit three cross sections. However, the most common type of Generalized Cylinder, which has a straight spine and a constant sweeping rule, can be specified efficiently using only four knots as illustrated in Figure 4. Due to the smaller number of required knots compared to the usual surface fitting and the local controllability in the stages of cross section fitting, Generalized Cylinder fitting can be done simply and efficiently.

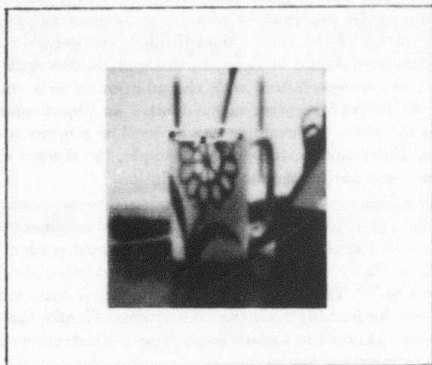


Figure 1: An example of fitting a cross section by three knots.

2.2. Using Pre-defined Parts Efficiently in Model Building

Some 3D objects have a hierarchical structure and have the same subparts. Once a part is described in terms of a Generalized Cylinder, the description can be used to define other parts efficiently. Besides, this has the advantage that 3D models can be described in compact form and modifications can be done to all the common parts simultaneously.

1) Identical part

The same definition of the Generalized Cylinder can be eliminated by the selection of an already defined part name. Also, if the relative position and the orientation of the part is identical to already defined parts, specifications of the relation described later are not necessary either. Otherwise, three knots are necessary

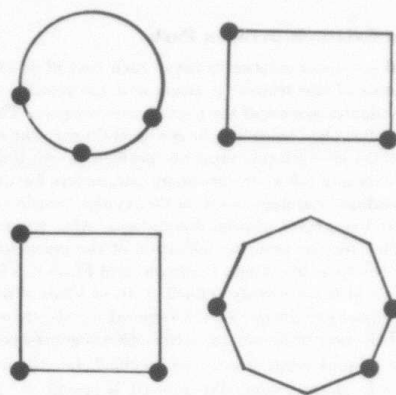


Figure 2: Cross section types

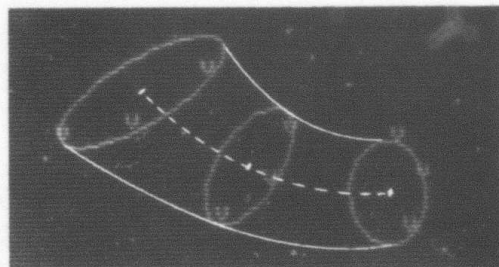


Figure 3: Fitting a Generalized Cylinder from knots.

for the specifications of the position and the rotation of its local coordinates system. Iterative definition of the identical parts enables the simultaneous definition of several identical parts. After the Generalized Cylinder definition of the first identical part and specification of the number of iterative parts, two or three knots are specified to determine the positions of the rest of the identical parts. Namely, if the positions of the identical parts are circular, three knots are required. If the position is linear, two knots are enough to determine the positions.

2) Symmetrical part

Some objects have symmetrical subparts. The symmetrical part can also be described in terms of the model descriptions already defined. For example, the port wing of the object jet aircraft is symmetrical to starboard wing. In this case, the specifications of port wing can be eliminated by utilizing the definition of the starboard wing which is symmetrical against the y-z plane of the fuselage. The model description of the port wing is generated from the descriptions the starboard wing.

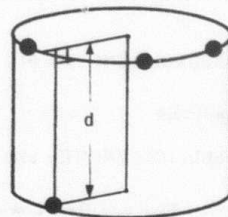


Figure 4: Fitting a Generalized Cylinder by four knots.

2.3. Spatial Relations between Parts

The final geometric relation between each part of an object is described in terms of the relative position and the relative rotation of the local coordinates system of the part to another part. The relations between parts described below can be specified through the menu selection. At least the affix relation must be specified to another part. The system automatically calculates necessary parameters for the transformation of coordinates system based on the relative location of the part in 3D space and generates relation descriptions. Also, several relations can be specified for the accurate definition of the geometric relation. For instance, relations like Align, Coplanar, and Flush can be specified to other parts which are already defined. If these kinds of relations are specified, the local coordinates of the Generalized Cylinder is slightly modified and the accurate model descriptions are generated.

Also the subpart relation must be specified, i.e. the name of the subpart must be defined when the subpart is specified. The system generates a subpart tree using the name. If the object example is not the first example of the class, the name can be selected from the menu.

3. Forming Object Class from Examples

After the model descriptions of the object instance are generated as described in Section 2, the model of the object class is learned inductively, based on the examples or models of the object instances.

3.1. Representation for Learning

In the ACRONYM system, data structures are represented by the frame-like structure, i.e., each data object is an instance of a unit. Units have a set of associated slots whose fillers define their values.⁴ Objects are represented by object graph whose arcs are subpart and affixment. The subpart arc describes a coarse to fine structural hierarchy represented by a subpart tree. Affixment arcs relate coordinate systems of objects. Class is represented through a mechanism of constraints and a restriction graph. The constraint is inequalities on algebraic expressions which defines a set of values which can be taken by algebraic expression in the slot, i.e., $5.0 \leq width \leq 6.0$. The restriction graph is used to organize the constraints into class, subclass and instance. Namely, a set of constraints can define object class, subclass or instance and the hierarchy of each set of constraints can be defined by restriction graph. Figure 5 shows an example of the class hierarchy of the jet aircraft. The nodes in each level have corresponding constraints, though one set of subpart tree, affixment tree, cylinder descriptions exists for the object class. The constraints of the predecessor are applied to its successor nodes, for instance, the constraints of B-747 are applied to both B-747B and B-747SP.

3.2. Rules of Generalisation

Generalization to form a class model can be done simply owing to the representation described above. The model descriptions of the object instance built by the system as described in Section 2 are written in terms of constraints("=" is one of the forms of the constraints). The constraints can be used for variations in size, in structure, and in spatial relationships. For example, a structure can be expressed like LEG-QUANTITY = 3. Generalization of the model descriptions can be done by generalizing these constraint. For example, suppose B-747B has the constraint,

$$FUSELAGE-LENGTH = 67.3$$

and B-747SP has the constraint.

$$FUSELAGE-LENGTH = 52.0$$

These constraints are generalized and the following constraint of the FUSELAGE-LENGTH of the class B-747 is obtained.

$$52.0 - \epsilon \leq FUSELAGE-LENGTH \leq 67.3 + \epsilon$$

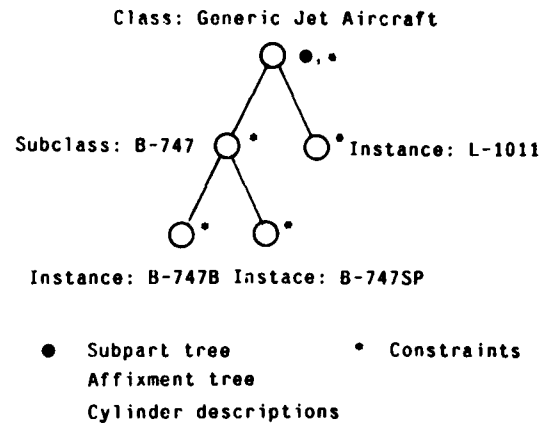


Figure 5: An example of the class hierarchy.

where ϵ is a tolerance. Constraints for structure can be generalized in the same manner. If the corresponding subpart cannot be found in a certain instance, the quantity of the subpart in that instance is regarded as 0.

In summary, the teacher teaches the object instance using an example and specifies the name of the class/subclass to which it belongs. The system generalizes the model descriptions of the object class as described. Namely, all the constraints in the predecessor nodes in the restriction graph are generalized. For example, if we add a new instance to the class B-747, say B-747X, the constraints in node 'B-747' and 'generic aircraft' are generalized.

The inductive learning described above uses only positive examples and the model descriptions sometimes become overgeneralized. The specialization of the class model through the negative examples or counterexamples should be done. In this system, descriptions of the class model can be specialized with the addition of new constraints. When the ACRONYM system misrecognizes an object which is not included in the class, the example can be used as a negative or counterexample. Using this misrecognized example, the teacher can add a new type of constraint in the model description.

Learning from examples can be categorized into two types, the one-trial, and the other, incremental. "Although this incremental method parallels human learning, it is apt to lead one down garden paths by an injudicious choice of initial examples in formulating the kernel of the new concept".⁷ Therefore, in this system, the teacher must select good examples for forming good class descriptions. Finally, the system is currently being planned to include some type of constructive induction in order to increase its power.

4. User Interface and Examples

The user-friendly interface of building models are described using actual examples in this section. Namely, the method to input knots in 3D space and to fit the Generalized Cylinder is described, based on the Generalized Cylinder fitting method described in Section 2.

4.1. User-friendly Interface

Figure 6 shows the following hardware devices which are used as user interface. (1) Scanning Stereoscope ODSS III (2) display device for stereo images (3) Voice Recognizer SYS300 (4) trackball (5) TSS terminal. The system generates 3D models through the following user-friendly interface using these hardware devices.

1) Convenient Input

The system is designed to use only voice and the pointing device. The keyboards are not usually used except in the case where names



Figure 6: Hardware of Stereo Modeling System

are specified. The voice input is especially useful in Generalized Cylinder fitting operations in which both eyes are fixed on an object and a hand is placed on the trackball as described later.

2) Menu Selection

Commands to the system are done in terms of menu selection using voice. With this menu method, users need not memorize commands and the operations can be learned easily, guided by the system.

Also, short help messages explaining what to do next are displayed in the lower part of the screen whenever necessary. If the user is completely lost as to what to do next, "help" in the menu can be selected. The system always prepares help commands in menus and the user can read displayed help messages.

4.2. 3D Input

The knot input is done directly in 3D space formed by stereo images. The position of the knot in 3D space can be calculated by the corresponding points in two different images based on the theory of Photogrammetry.⁵ From the viewpoint of speed and accuracy, 3D input is done through the following two stages.

1) Coarse initial positioning with two cursors

The coarse initial position of the knot is specified by pointing to corresponding points in the stereo images, using two cursors displayed on the same epipolar line⁶ in the stereo pair. Although the 3D positioning using the cursor moved by the trackball is fast, it is impossible to specify the position between one pixel. The one pixel disparity between the corresponding points in stereo images sometimes affects the accuracy of the depth measurement significantly. Therefore, in order to position the knot in 3D space with a high degree of accuracy, the following 3D pointer, which can be moved in 3D space with any degree of accuracy, is used for fine positioning of the knot.

2) Fine positioning with 3D pointer

The accurate 3D position of the knot can be specified through the 3D pointer displayed in 3D space. The 3D pointer can be moved in six directions in 3D space with voice commands, as illustrated in Figure 7. Also, the speed of the 3D pointer can be changed, i.e., the speed can be changed to $\times 2$ and $1/2$ by "fast" and "slow" commands respectively. For instance, if you say "slow" three times, the speed is reduced to $(1/2)^3 = 1/8$. The 3D pointer can be moved in 3D space after all the initial input of necessary knots for generating a cross section or a Generalized Cylinder is finished. Although it is difficult to display the 3D pointer between pixels in the image, the 3D position of the knot can be determined with

great accuracy by observing a displayed cross section reflecting the move of the knot. For example, a radius of a circular cross section is sometimes highly sensitive to the slightest movement of one knot.

4.3. Stereo Images

Stereo images of objects are obtained by placing objects in front of a stereo camera. Currently, the online stereo camera is not used and stereo images which are stored in a disk file are used by the system. As described before, the one pixel difference is sometimes very important for accurate Generalized Cylinder fitting in 3D space, so a high-resolution display device is needed. To solve this problem, we built zoom function, i.e., a part to be modeled can be enlarged and displayed whenever accurate modeling is required. This zoom function enables accurate modeling, as if a high-resolution display device were available. Also it is better that the stereo images themselves should be high-resolution. For this purpose, in addition to the normal size picture, high-resolution stereo images are also used by the system. Any portion of the stereo images with any resolution are displayed using the resolution pyramid of the stereo images with this zoom command.

4.4. Examples

The user-interface described above will be shown with actual examples of Generalized Cylinder Fitting operations. Figure 8 shows a displayed original stereo pair of industrial parts with a command menu. Suppose we built a model of one of the parts. After the zoom command is selected and the location of the zoom window is specified with the trackball, the enlarged stereo pair of the selected part is displayed as shown in Figure 9. With two cross-haired cursors, the initial 3D position of a knot is specified as shown in Figure 10 after the type of Generalized Cylinder is selected from the menu. In this figure, "v" shows the position of already specified knots. After the specifications of three knots, the knots can be moved by selecting one knot with the cursor pointing to the vicinity of the knot. The 3D pointer "!" appears as shown in Figure 11 and the 3D pointer can be moved in 3D space with voice commands. The displayed cross-section shown in Figure 11 is also moved/transformed, reflecting the movement of the 3D pointer. After the fitting operation of the cross section is finished, the last knot, in this example, is specified. When all the knot input is done, the Generalized Cylinder is displayed and the knots can also be moved at this stage. Figure 12 shows the displayed Generalized Cylinder in terms of two cross sections. As this system does not use any special graphics hardware and since it is written in Lisp called Successor Lisp (also some graphic packages are written in C), it takes too long for real-time interaction if moving/transforming Generalized Cylinders are well displayed with hidden-line removal. Therefore only cross sections are displayed

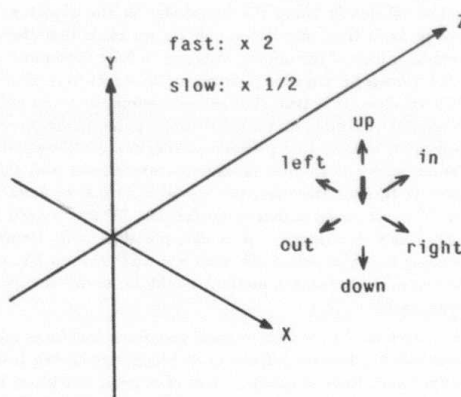


Figure 7: 3D pointer controlled by voice commands

instead of a Generalized Cylinder while the Generalized Cylinder is being fitted. Besides, hidden-line removal is not necessary as long as the stereo line drawing is used as there is no ambiguity of 3D structure in the stereo line drawing. Figure 13 shows Generalized Cylinder descriptions generated by the system after all the fitting operations are finished.

Although it might be accurate enough for most purposes (The actual size of the part is 31mm×33mm×16mm), the Generalized Cylinder descriptions would have been more accurate if we had measured camera parameters more accurately.

It takes about three or four minutes to fit one Generalized Cylinder with four knots, though the problems of display time and 3D input accuracy had to be solved. The total time required to fit Generalized Cylinders to all the subparts of an object depends on how many subparts it has. For example, if the object is formed by two or three subparts, ten minutes would be enough for building the model of the object. With this user-friendly interface, even a person who is not a computer expert can learn how to build models easily and can build them simply and quickly without errors.

5. Conclusions

The Stereo Modeling System which enables simple, fast model building was described. The key ideas here are the following: (1) build a model of an object instance from a 3D example of the object. (2) learn the class model of the object from examples. This system is easy to learn, i.e., even a person who is not a computer expert can master it in an hour or so and models of 3D objects can be built easily and quickly, e.g., a model of a simple object with two or three parts can be built in 10 minutes without errors.

Although the problem of the accuracy of the modeling was solved by the zoom function and the 3D pointer which can be moved in 3D space within one pixel of the stereo images, this modeling system might not be suited for the highly accurate modeling of complicated objects. However, the simplicity rather than the high accuracy of 3D model descriptions is important in Image Understanding. Even we don't have accurate 3D models of objects in the world in our brains. Although the 3D models which this system produces might not be suited for CAD or Computer Graphics in which a high degree of accuracy is important, they are believed to be accurate enough for most applications in image understanding.

It would be ideal if 3D models were built automatically without any aid. However, it is still difficult to build useful 3D models for Image Understanding fully automatically. In this system, the example of the object is divided into parts according to the required accuracy based on human judgement and geometric features of the object are taught and described efficiently using the knowledge of the object we have. Models can be built from one stereo pair for we know that there is no tail in the hidden part of the object. However, a fully automatic system without such knowledge requires pictures of the object from all different angles. Nevertheless, it is true that in our system, a stereo pair from a good viewpoint is needed and several stereo pairs might be required for a complicated object. It is possible to continue building the model from different angles if camera parameters are known and this may be necessary in future. However, the key idea here is to build models directly in 3D space using actual examples, i.e., 3D space need not be stereo images and stereoscope. It is also possible to fit Generalized Cylinders using knots in actual 3D space via a 3D pointer like a robot arm. This type of alternative method might be useful if a good 3D pointer is available.

Some classes of objects have various geometric instances and they are best described by functional features and higher geometric features.¹ On the other hand, there is another class of objects which can best be described by geometric features and is difficult to describe by functional features, e.g., there is no man with 10 legs and what are the functional features of a man? Also there is a class of objects which can be described both functionally and geometrically, e.g., tennis balls are always round, notebooks are rectangular, and they have functions. As

the first step towards the powerful modeling system which can handle both functional and geometric features, the geometric modeling system described here is designed to build models of objects which can be described by geometric features. This system and the ACRONYM system do not yet handle functional and higher geometric features. They would become more powerful if, in future, they could handle these features.

References

- [1] Brooks, R.A., *Symbolic Reasoning Among 3-D Models and 2-D Images*. Ph.D. Thesis AIM-343, Department of Computer Science, Stanford University, Stanford, California, June 1981.
- [2] Baumgart, B.G., *Geometric Modeling For Computer Vision* Ph.D. Thesis AIM-249, Department of Computer Science, Stanford University, Stanford, California, October 1974.
- [3] Binford, T.O., *Visual Perception by computer*. Proc. IEEE Conf. on Systems and Control, Miami, December 1971.
- [4] Newman, W.N. and Sproull, R.F., *Principles of Interactive Computer Graphics* McGraw-Hill, New York, 1979, pp. 154-158.
- [5] Slama, C.C. (ed.), *Manual of Photogrammetry*. American Society of Photogrammetry, Maryland. 1980
- [6] Baker, H.H., Binford, T.O., Malik, J., and Meller, J.F., *Progress in Stereo Mapping*. Proc. Image Understanding Workshop, Arlington, Virginia, June 1983.
- [7] Michalski, R.S., Carbonell, J.G., and Mitchell, T.M., *Machine Learning*. Tioga Publishing Co., Palo Alto. 1983.
- [8] Winston, P.H., Binford, T.O., Katz, B., and Lowry, M.R., *Learning Physical Description from Functional Definitions, Examples, and Precedents*. National Conference on Artificial Intelligence, Washington, D.C., 1983, pp. 433-439.

Acknowledgement: This work was supported in part by the Air Force Office of Scientific Research under contract F49620-82-C-0092.

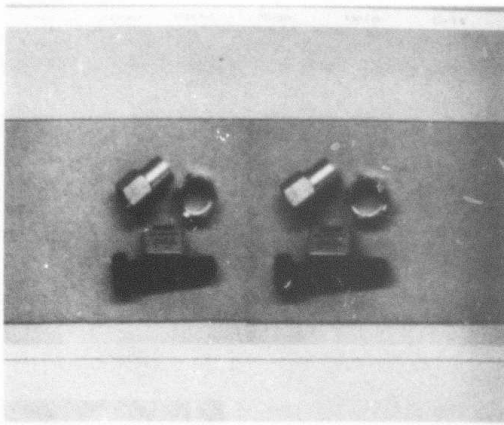


Figure 8: Stereo pair of industrial parts(original size)

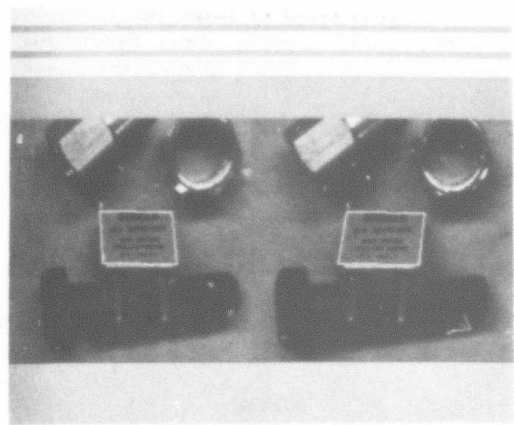


Figure 11: Fitting the cross section with the 3D pointer.

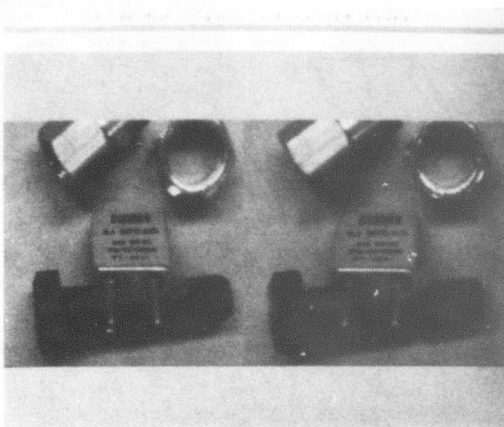


Figure 9: Enlarged stereo pair.

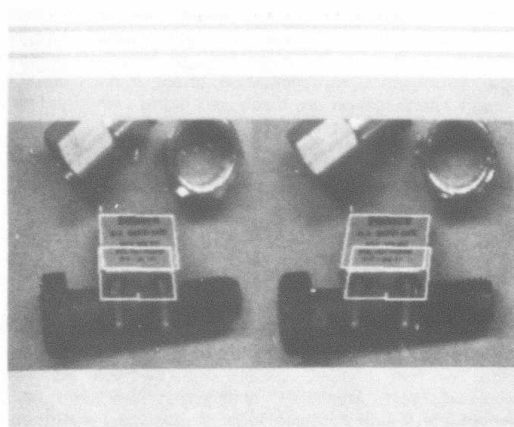


Figure 12: Fitting the Generalized Cylinder.

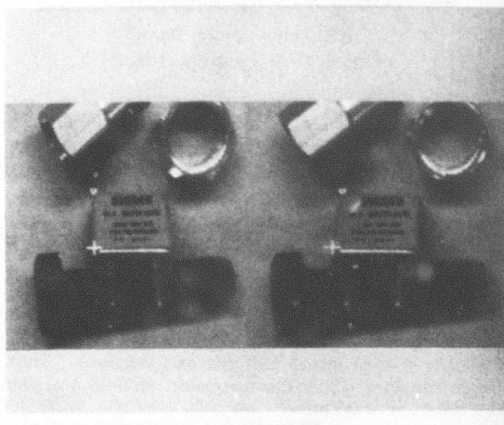


Figure 10: Initial knot input with two cursors.

Parts Descriptions		
Single-cone having		
cross-section	type	rectangle
	width	= 30.871682
	height	= 32.732823
spine	type	straight
	length	= 17.251298
sweeping rule	constant	
x0	= -1.958054	
y0	= 2.922167	
z0	= 533.889064	

Figure 13: Generated part Descriptions.

ACRONYM MODEL BASED VISION IN THE INTELLIGENT TASK AUTOMATION PROJECT

David M. Chelberg, Hong Seh Lim, Cregg K. Cowan

Stanford Artificial Intelligence Laboratory
Stanford University, Stanford CA 94305

Abstract

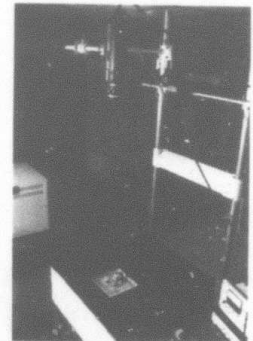
The Intelligent Task Automation Project involves the automatic location and assembly of a tray of parts in an uncontrolled environment (i.e., no control of lighting). This paper provides a brief description of the ACRONYM vision system and concentrates on extensions for location of the assembly components. The characteristics of the components are significantly different than the domains in which ACRONYM has been demonstrated - in particular the parts include holes and springs. Problems in representation, image feature prediction and image interpretation are discussed along with significance for future work in model-based vision.

Introduction

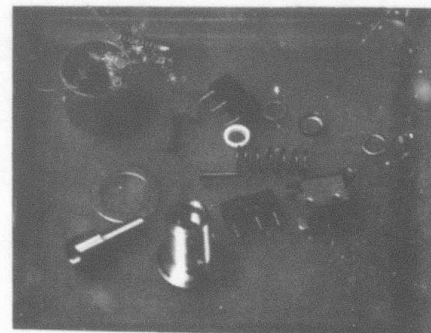
The ITA Project

The Intelligent Task Automation (ITA) Project¹ includes the automatic assembly of a push-button micro-switch in an unstructured environment. That is, no special lighting or part-feeders are used. The ITA project is nearing the completion of the first of two phases: phase one involves technology development and feasibility analysis; phase two is an implementation which achieves goals of speed and performance. The system is planned to operate as follows:

- Off-line strategy selection
 - Analyze a three-dimensional model of each component of the micro-switch and predict the shapes and relations between shapes which will be seen in a grey-scale image.
 - From the predicted features and relationships for each part, select a set of particularly important characteristics.
 - Select algorithms to locate the expected features, and to match the relations.
- On-line execution
 - Receive a tray of parts (assume the tray contains all components for one micro-switch).
 - Shake the tray to eliminate unstable part poses.
 - Locate the first (next) part for the assembly in order. Send the part position and orientation to a robot arm controller for part acquisition and assembly.



Grey-Scale Camera and Tray of Parts



The Switch Assembly Parts

Figure 1: Camera and Tray

Two types of sensors are available for part location - a grey-scale CCD camera is mounted above the tray and there are two range sensors. One is a sparse sensor mounted on an arm, and the other is a dense range sensor mounted overhead -- both using a plane of laser light in a triangulation process.

The ACRONYM vision system

The ACRONYM vision system^{2,3} is a feasibility demonstration of automatic feature prediction and image interpretation. ACRONYM was chosen for its ability to "reason" about three dimensional geometric models, and from this analysis, to automatically predict image features and relations. Thus, no training is required - only the geometric models are given to ACRONYM. In this paper, we will describe ACRONYM's capability to reason about a new class of objects (the micro-switch parts) and to recognize these parts from grey-scale image data.

ACRONYM was initially used for location of aircraft in aerial photographs. It was also used to model short range views of some simple electric motors. Hughes Research Laboratory then used ACRONYM to locate ships from various camera angles. The ITA switch parts represent a significantly different set of geometric features than these initial domains.

The Base ACRONYM System

The base ACRONYM vision system reasons about models of the objects which may appear in an image and is domain-independent (i.e., the reasoning is based on geometry rather than on domain-specific knowledge). The system can be divided into four modules - modeling, prediction, image description and image interpretation. The user models the objects in the domain and specifies the relationships between the objects and in the world. Base ACRONYM analyzes the object and world models to predict image features and relationships. When given an image, the interpretation module uses these predictions to identify objects in the world image, inferring three-dimensional information concerning the shape, structure, location and orientation of the objects.

Modeling System

Objects An object model describes the three-dimensional shape and structure of an object. The object is divided into simple subparts, each described by a generalized cone volume primitive. (A generalized cone describes a volume by sweeping a planar cross-section for example, a circle, along a spine⁴.) Base ACRONYM implements a subset of the possible volumes represented by generalized cones. The spine must be straight and cross-sections may be one of a circle, square, or rectangle. The cross-section may be deformed linearly in one or both dimensions (i.e., the dimensions of the cross-section) and may be held at an arbitrary fixed angle to the spine. For example, a cube is a square swept along a straight spine. This representation is not unique; each of the three axes of symmetry may be used for the spine in this example.

In addition to individual objects, base ACRONYM can represent generic classes of objects. This is accomplished by using symbolic, rather than numeric, parameters for the volume descriptions. In the initial ACRONYM demonstrations, 747 and L1011 aircraft were described by the same generic model containing symbolic parameters. Each aircraft type had a set of numeric constraints on these symbolic variables.

The structure of an object is described by the relative positions and orientations of its subparts. The volume of each subpart is described by a generalized cone and each generalized cone has a local coordinate system. The object also has a coordinate system. The position and orientation of a subpart (generalized cone) relative to the object is specified by the transformation between their coordinate systems.

Scene An object which is expected to be seen in an image is placed into the world model by specifying the transformation between the world and object coordinate systems. Usually some or all of the transformation parameters are symbolic - if the position and orientation of an object are known in advance, then a vision system is not needed to locate the object.

Camera In addition to knowing which objects may be in the world image, base ACRONYM uses knowledge about the camera and the camera location relative to the world coordinate system to guide its prediction and interpretation. In the ITA task the camera location is precisely known but ACRONYM can reason about the appearance of objects without precise information. For the aerial photo demonstrations, the focal ratio of the camera

was known exactly but the altitude and camera orientation were underspecified. The altitude was constrained to be between 1,000 and 12,000 meters, and the camera was known to be pointing generally at the ground.

Predictions

Given a set of object models and a scene and camera model, geometric reasoning rules are used to predict shapes and relationships that will be observable in an image. Features which are always observable (observable within the entire modeled range of camera viewpoints) are called invariant. Unfortunately, few features, in general, will be visible over the entire range of camera and object position/orientation. Usually there are accidental viewpoints in the range of camera and object locations. For example, the sides of a cylinder will produce parallel lines in an image for all viewpoints except when the viewpoint is collinear with the axis of the cylinder (not invariant). However, if the cylinder is known to lie on its side, and the camera position is known to be above the plane of the cylinder, then the parallel lines are invariant.

ACRONYM also analyzes the range of camera/object geometry to predict features and relations that change slowly with changes in viewpoint, called quasi-invariants, but which are not invariantly observable. If no knowledge were available in the previous example, the length of the cylinder would still be quasi-invariant because it changes slowly with change in viewpoint.

Predictions are described by the prediction graph. The nodes of the graph represent the predictions of image shape features, and the arcs of the graph specify relations between the features. As will be seen, relationships between shapes are critical to ACRONYM.

Shape Prediction ACRONYM predicts image shapes for the "swept" sides and for the end faces of a generalized cone. A trapezoid is a two dimensional projection of the "swept" sides of a generalized cone with a straight spine. The projection of an end face can also be described by a trapezoid for square and rectangular cross-sections. Circular cross-sections produce ellipses in an image. The size of the shapes is constrained by upper and lower bounds based on the modeled range of camera-to-object geometry.

Relationship Prediction In addition to shapes, relations (arcs of the prediction graph) between shapes are also predicted. During interpretation, pairs of hypothesized matches of image features to prediction nodes (shapes) are checked for consistency by attempting to find the predicted relationship between them (which is represented by a prediction graph arc). Prediction arcs are generated to relate multiple shapes predicted for a single cone as well as between different generalized cones (subparts) for an object. The latter are actually more important in arriving at a consistent global interpretation of collections of image features as complex objects.

Exclusive arcs relate image features which are mutually exclusive. For example, the two end faces of a solid cylinder cannot be visible at the same time. Collinear and connected arcs represent invariant relationships. If two straight lines are collinear in three space, then their images must be collinear, except in the rare case of a single degenerate point. If two cones are physically connected in three space, they are connected in any image. Angle arcs are predicted between the spines of two generalized cones. For those image features which are not connected, distance arcs are generated to provide additional constraints. Finally, contained arcs relate two predicted shapes, one of which contains the other in the image.

Back Constraints The shape and size of image features are clearly viewpoint dependent. The predictions are constrained with upper and lower bounds based on the range of camera to object geometry. In addition to predicting the shapes and relations, the prediction module also generates equations to constrain the relative camera-object geometry from image measurements. For example, from most viewpoints the circular end of a cylinder will appear as an ellipse. Furthermore, the eccentricity of the image ellipse constrains the orientation of the cylinder relative to the camera.

Image Description

ACRONYM describes an image as a graph (the picture graph) of trapezoids and ellipses. That is, edges must be linked into trapezoids and ellipses. The prediction graph provides general guidelines for the shapes and sizes that can be expected from the image. The description module then looks for those shapes, ignoring the predicted relationships between them. The result is a picture graph, consisting of shape descriptions and their locations and orientations in the image.

Interpretation (Matching)

ACRONYM interprets images by subgraph matching (subgraph isomorphism) between the picture-graph description of the image and the prediction-graph expectation. It proceeds by combining local matches of shapes to individual generalized cones (from the ribbon finder), into global matches for complete objects. The global interpretation must satisfy the requirements specified by the arcs of the prediction graph (i.e., the relationships between subparts). The constraints that each local match implies on the three-dimensional model must be globally consistent. As hypothesized local matches are combined into more complete objects, the size of image shapes and relationships provide constraints on the camera position and orientation relative to the object. Inconsistent constraints indicate an incorrect hypothesis.

Initial Domain (i.e., Aerial View of Aircraft)

The success of the original implementation is largely due to the fact that most objects exhibit structure which is well approximated by skeletal description. Consider the human ability to recognize stick figure drawings. Stick figure drawings resemble the objects they represent only in skeletal structure - the relative lengths of each segment and the relationships between segments⁵. Aircraft in particular are skeletal and can be disambiguated by their skeletal structure.

Extensions to ACRONYM

The application of ACRONYM to industrial parts represents a significant domain shift. The base ACRONYM system included sufficient geometric reasoning to demonstrate the power of feature prediction and image interpretation but the implementation was not completely general. In particular, the features of aircraft from aerial viewpoints produce image trapezoids. Not surprisingly, a large proportion of its geometric reasoning rules are devoted to prediction and interpretation of trapezoids.

In order to recognize the parts of the switch assembly, modifications to base ACRONYM were required. These modifications included changes to and extension of the modeling, prediction and image interpretation (object matching) modules. In this section we discuss the modifications which were necessary and the performance of the ITA version of ACRONYM on the micro-switch components.

Modeling

Holes Base ACRONYM was capable of reasoning about solid volumes only. Every item in the switch assembly contains a hole and for many objects the hole is an essential characteristic (e.g., a washer). Our description of holes follows the approach of base ACRONYM for solid objects. We describe a hole by a generalized cylinder with an additional tag indicating that it is void (negative volume). We restrict the representation to holes which are surrounded by some solid. This restriction allows us to model all the parts in the ITA task but avoid arbitrary intersection of volumes.

Springs While many man-made objects are easily described by a generalized cylinder with straight spine, the micro-switch assembly includes three springs (two cylindrical and one conical spring). We have implemented a new generalized cylinder spine type: the helix. A spring is a circular cross-section swept along a helical spine. A helical spine can also be used to describe the threads of a bolt by using a triangular cross-section. Special primitives describe the ends of a spring. A squared end indicates that the helix is deformed (squared) on the last revolution of each end. A plain end is simply the undeformed helical spine. The model of the conical (tapered) spring uses the taper primitive to describe the ratio of the small end of the spring to the large end.

Stable States A further extension to the modeling system is a representation for the stable states of an object. A stable state is an orientation in which the object will remain stable on a level surface⁶. For example a convex regular polyhedron is stable on any of its faces. A right circular cylinder is stable on its ends and on its side (its stability depends on the ratio of radius to height). The representation consists of constraints on the orientation and position of the object in the tray. Some information may be unspecified, such as the orientation about the right circular cylinder's axis (i.e., it can roll when lying on its side). By using symbolic constraints we can not only model the nominal stable state position (on a plane), but can also represent perturbations of the stable state explicitly. For example, in its nominal stable state, the large washer lies flat on a plane. Our representation for it, however, can also describe its orientation when lying at an angle (on top of another object).



Nominal Leaning
Figure 2: Stable States of a Washer

While ACRONYM is capable of reasoning about positions and orientations which are completely unconstrained, the stable states allow ACRONYM to produce tight bounds on the predicted shapes and relationships. Stable state constraints reduce the complexity of prediction and allow image interpretation to proceed more quickly and to achieve higher confidence object hypotheses.

The position of the camera relative to the world coordinate system is well known. When combined with the stable state description for an object, this means that the predicted shape and relationship parameters are constrained to exact values (i.e., lower and upper bounds are equal) and no back-constraint equations are generated. However, if the object orientation is described with some error tolerance (in case the object may lean on another object), back constraints will be useful in determining the actual object orientation.

Prediction

The prediction module received the most extensive changes. We improved ellipse prediction, and added a variety of new relational predictions between shapes. ACRONYM finds image shapes (which match shape predictions) and uses predicted relationships to prune the search for a set of shapes which is consistent with the hypothesized object. Increasing the number and strength of relationship predictions improves the performance and accuracy of image interpretation.

Relationships between the subparts of an object are very important to ACRONYM's performance. For the ITA parts we found that the relationships which base ACRONYM could predict were not sufficient. Stronger relations were needed to constrain possible interpretations and to link the different shapes predicted. Because base ACRONYM had no prediction for holes, no relationship existed between the inner void and outer solid of a hollow cylinder. (Base ACRONYM assumed everything was solid so it never reasoned about the insides of anything.) Washers would have been predicted as a single ellipse. Clearly these weak predictions, such as a single ellipse for a washer, would not lead to high confidence in object interpretation hypotheses. Thus we needed new relational invariants, or at least semi-invariants. The extensions that are most important for micro-switch parts follow:

- reasoning about stable states
- reasoning about holes
- concentric relation between concentric cylinders (e.g., solid and void cylinders describing a washer)
- connected relation between ellipses and trapezoids and between ellipses and ellipses
- parallel relation between coils of a spring
- enclosed relation between coils of a spring and the imaginary bounding of the spring

Figure 3 demonstrates prediction of an object with holes. This prediction example is for stable state three of the bushing (standing on its smaller end). The bushing is composed of five cylinders - two outer (solid) volumes and three inner (hole) volumes (see figure 3a).

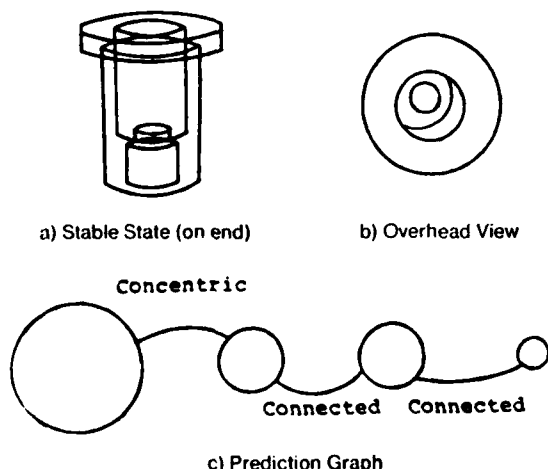


Figure 3: Simplified Prediction for Bushing

Stable state reasoning Reasoning was added to achieve better prediction about the range of shapes that will be visible in an image. The constraints on object location form a context for the object model. ACRONYM was modified to reason about the appearance of an object model within multiple contexts (i.e., multiple stable states). The improvements allow greater performance and produce more accurate image shape descriptions.

Holes We have implemented some useful, although certainly not complete, reasoning about holes. We assume that a hole is fully contained in a solid - the shape of the hole and the shape of the solid are the only shapes to consider. We need not consider arbitrary intersections. (Note that generalized cylinders are not closed under arbitrary intersection). While this may seem restrictive, the solid-hole relationships in most industrial parts are quite simple - collinear, parallel or perpendicular axes.

Much of the reasoning about which surfaces of a solid may be visible does not apply to holes. Surfaces of a solid generalized cylinder may occlude surfaces which are further from the viewpoint. The touching surfaces of connected solids are also occluded. Holes, however, are invisible. A hole cannot occlude anything - neither its own surfaces which are further from the camera nor any object connected to the hole. The swept surfaces of a solid object are often quite important indeed, for aerial views of aircraft, only swept surfaces are visible. The swept surface is almost never visible for a straight hole. While it is possible to imagine an object where the swept side of a hole is visible (e.g., half of a hole), we do not consider such rare objects.

Concentric Shapes The concentric relationship is important for many of the objects which contain holes, especially for washer-like and ring-like objects. If two generalized cylinders are concentric in three dimensions, they are concentric in any projection. That is, concentricity is invariant and therefore a very strong pruner of hypothesized interpretations. For the ITA project, the concentric relationship is predicted for 10 of 15 objects (17 of the 32 stable states).

A weaker form of concentricity was also implemented. If two cylinders (of unequal length) share the same axis and the shorter one is inside the other, then whenever you see both holes, they are concentric, or nearly concentric. This applies, in particular, to the case of structures inside a cylinder - such as the bushing viewed from either end.

Connected Shapes Reasoning about the appearance of holes is often different than for solids. Both ends of a solid are never simultaneously visible (the swept side occludes one end). For a hole, however, both ends are visible whenever the viewpoint is near the line of the hole axis. Furthermore, if both ends are visible then they must be touching in the image (remember the hole is surrounded by some solid). The ends of a hole are connected.

While some code to predict ellipses existed in the original ACRONYM system, ellipse prediction and matching were not fully implemented - few relationships involving ellipses were implemented. We added a stronger form of connectedness to ACRONYM. Instead of connectedness between only trapezoid and trapezoid, we now predict connectedness between ellipses and trapezoids, and between ellipses and ellipses. This relationship is important, for example, for viewpoints from which one sees the side and one end of a cylinder.

Parallel Trapezoids Relation between parallel trapezoids is important for the prediction of springs. If two generalized cylinders or two parallel coils of a spring are parallel in three

dimensions, their images are also parallel under any projection. This relation is another invariant we have made use of. When the spring is lying flat on its side, in one of its stable state, the parallel coils of the spring appear as a set of parallel trapezoids in the image (figure 4). These trapezoids are not connected in the image. Since the spring is modelled as a single part and not as a collection of subparts, the distance, angle, and contained relations are no longer useful. A new parallel relation is thus predicted between each pair of the trapezoids. This new relation also helps to prune off other undesirable random trapezoids in the image.

Enclosed Trapezoids To increase our confidence in prediction of springs, we also predict an imaginary bounding hollow cylinder enclosing a spring (figure 4). The coils of the spring are enclosed in this imaginary cylinder. Thus, when the spring lies on its side, the projection of this imaginary cylinder will appear as a trapezoid enclosing all the parallel trapezoids projected from the parallel coils of the spring. When the spring is standing on its ends, the prediction is exactly the same as that for a hollow cylinder of the same size.

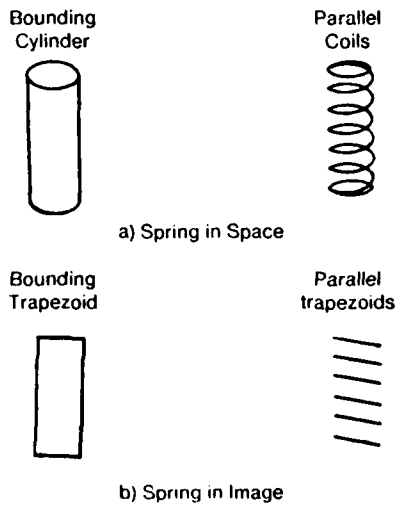


Figure 4: Spring Features

Performance on the ITA parts

Our changes have been very successful. Of the fifteen parts, ten can be recognized in any stable state. We can automatically generate shape predictions for all of the remaining five parts but the currently implemented relationships are not sufficient to link the shapes (i.e., we cannot yet locate these two parts in an image).

The ITA version on ACRONYM has been tested on data from real pictures. We have fully automated the process from picture to interpretation; the linking of edges into image shapes is accomplished automatically (the original image description module did not find ellipses).

Although analyzing an object model and predicting the image shapes which it will produce is expensive, it need only be done once. That is, predictions are done "offline" and stored for later use by the image description and image interpretation modules. The complexity of the object increases the cost of prediction. The large washer, a very simple object, requires 14 CPU seconds on a Vax 11/780 while prediction for the switch element requires about 6 CPU minutes for both of its stable states. A large performance improvement can be realized by recoding the

algorithms; the current implementation uses a MACLISP compatibility package which runs on top of FRANZ LISP.

Figure 5 illustrates ACRONYM's performance at image interpretation. Although the parts will be located individually for the actual assembly, for this example we instructed ACRONYM to search for the bushing and for the two switch elements simultaneously. Only the strongest hypothesis for each object is shown in the interpretation. Notice also that the image data is imperfect and that one terminal pin is missing. In general, deformed or missing subparts of an object affect only the strength of the interpretation hypothesis and will not cause a hypothesis to be discarded.

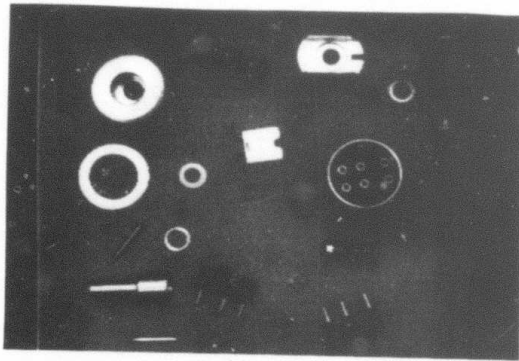
The asymmetry of the switch element is an important consideration for the assembly operation. ITA ACRONYM has determined that both switch elements lie on their "left" side. Two incorrect hypotheses were also generated, each labeling a switch element with the wrong stable state (i.e., lying on its "right" side). The incorrect hypotheses contain fewer predicted shapes (they are weaker) because each hypothesis omits the middle pin (the middle pin position is inconsistent with the right-side hypothesis). Generation of the switch element hypotheses required 85 CPU seconds in this example (again, improvements can be achieved by re-coding).

When searching for the bushing in any of its three stable states, many of the concentric shapes in figure 5c caused ACRONYM to generate bushing hypotheses. The correct interpretation is the strongest (figure 5d) - all four image ellipses are consistent with the stable state 3 hypothesis. The same ellipses also form the second strongest interpretation; three of the ellipses are consistent with the bushing standing on the other end. That is, the image shapes from the bushing generated the two strongest interpretations.

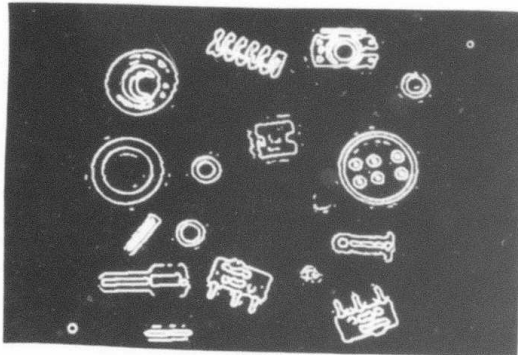
Several weak hypotheses were generated from other sets of image ellipses. For example, ACRONYM generated a weak hypothesis that the large washer (the concentric circles just below the bushing) is a bushing standing on end. The outer diameter of the washer is identical to the bushing and the inner diameter is within error tolerance. The correct hypothesis for the bushing is stronger because more of the predicted features were located in the image (i.e., the internal structure of the bushing disambiguates the possible interpretations).

Sometimes ACRONYM is unable to generate a single strong hypothesis for an object. For example, if the viewpoint in the preceding example had been oblique to the point that no internal structure was visible, the incorrect hypothesis (the washer) would have been equally strong. Better object identification can sometimes be achieved by tightening the error bounds on finding an ellipse (e.g., to eliminate the inner diameter of the washer). However, a few objects produce ellipses of identical dimensions and cannot be uniquely located by finding image shapes. When no one object hypothesis is sufficiently strong, the sparse range sensor (light stripe) will be used to obtain a depth profile to distinguish between the hypotheses.

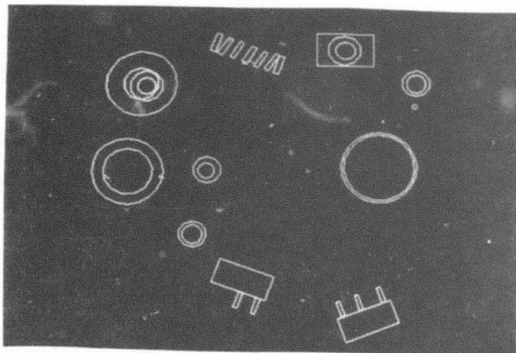
Figure 6 illustrates the situation where ITA ACRONYM fails to detect the desired part. The bottom switch element shapes conform to the poor edge data for this image. The shapes for the top switch were artificially degraded for illustration. Because image data is inexact, image interpretation allows for errors (in both the base and the ITA versions of ACRONYM). In this example, the image trapezoid for the body of the lower switch element is within error bounds, but the body of the upper switch is not (it does not match the prediction). Only three of the six terminal pins match the shape predictions (the upper two pins of the lower switch element and the leftmost pin of the upper switch element). Image errors also affect the relationships between



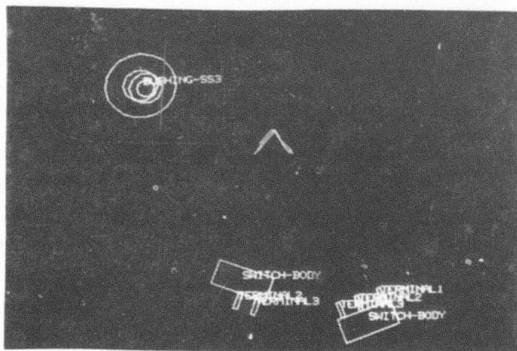
a) Image



b) Edges

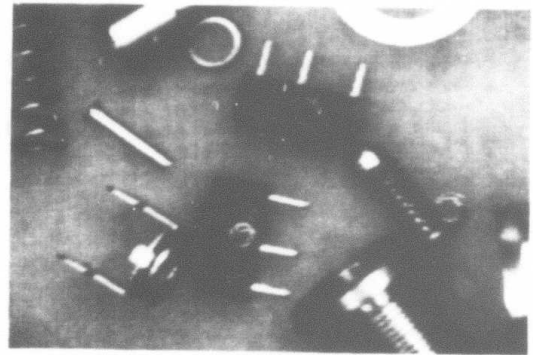


c) Shapes

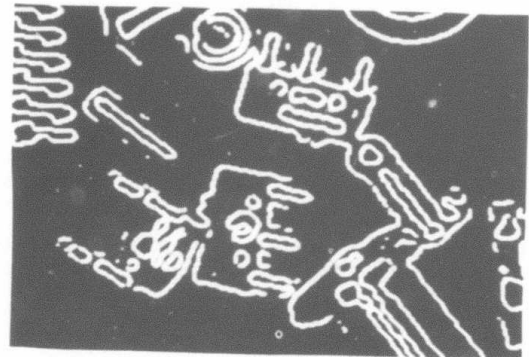


d) Interpretation

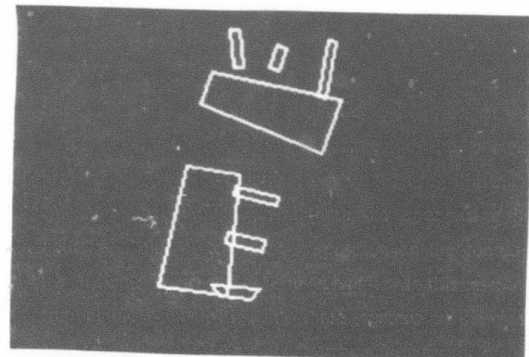
Figure 5: Search for Bushing and for Switch Element



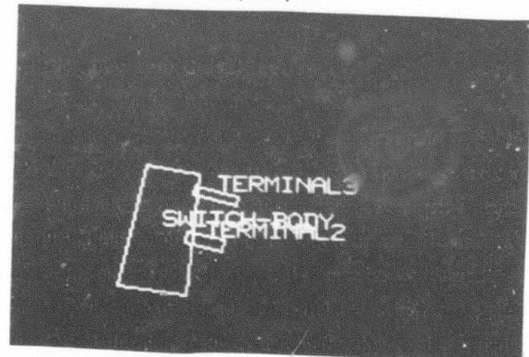
a) Image



b) Edges



c) Shapes



d) Interpretation

Figure 6: Search for Switch Element with Rad Data

shapes. Even if the upper switch body had matched the shape prediction, the angle between the body and the leftmost terminal pin is out of the allowed error tolerance (the hypothesis would be discarded). For any composite object (multiple subparts), image data must match at least two subparts. Three shapes for the lower switch element are within the error tolerance and their relationships are mutually consistent, that is, the image data is sufficient to form and retain the displayed hypothesis.

Future

Planar Reasoning

ACRONYM currently reasons about 3-D shapes and 3-D relationships (such as collinearity) which are invariant when projected into two dimensions. This type of reasoning provides useful constraints on viewpoint but is inadequate for detailed object recognition or inspection. The housing (figure 7) contains a plane with 6 holes in a pattern. While it may be possible to identify 3-D relationships for this pattern of holes, this is essentially a collection of 2-D relationships and reasoning in two dimensions is more appropriate.

Notice that the hole pattern is not symmetric about the vertical axis: the orientation of the object is determined by recognizing the orientation of the hole pattern. We will describe the planar cluster of holes by specifying a coordinate frame for the plane and describing the location of each hole in that coordinate system. The predictions will relate size of the circular plane (containing the 6 holes) to the size of each hole and will specify that the major axes of the image ellipses must be parallel.

With the addition of this 2-D reasoning, ACRONYM will predict a hierarchy of features -- "coarse" 3-D features for generating object hypotheses, and "fine" 2-D features for detailed reasoning. When interpreting an image, the hypothesize

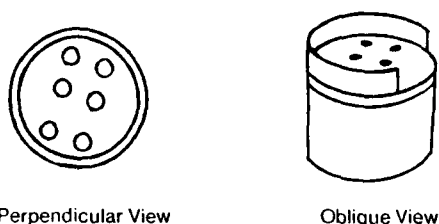


Figure 7: Housing

and test strategy will first match coarse image shapes to 3-D predictions, then fine image features to the 2-D predictions. Reprocessing of the raw image may be required to obtain the desired line image data (over small portions of the image).

Reflectivity

Specular reflections are particularly important for the ITA parts (as for any shiny object). ACRONYM currently contains a very simple light-source model (a single point) which was used for crude shadow prediction in a previous project. Ideally, we want to reason about multiple and more complex light-sources to predict characteristics of specular reflections.

For example, a shiny cylinder usually exhibits bars or stripes. The specular surface approximates a mirror. The cylindrical shape of the mirror causes light from a large area of space to be reflected onto a small area of the viewing plane. Under general lighting, many large light sources (white walls, fluorescent lights, etc.) produce obvious light stripes. Each stripe is widened (diffused) by the imperfections of the mirror, i.e., the shiny

surface is not a perfect mirror.

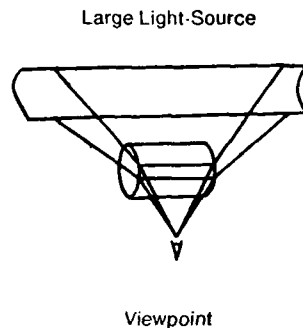


Figure 8: Light Stripes on a Metallic Cylinder

Reasoning about light-sources can also predict shadows. In a number of images, a cylindrical object reflects its own shadow. This shadow is important as it often masks the actual boundary of the cylinder. We wish to account for this effect when calculating the upper and lower bounds on the apparent diameter of the cylinder (when lying on its side).

Useful predictions can be achieved without complex light-source models. Under the general lighting assumption, a shiny cylinder will always exhibit stripes. Stripes will be visible anytime the portion of the environment which is reflected by the cylinder is not uniform. Predicting the number, size, or location of the stripes requires sophisticated light models.

Successor

Part of the value of this project is that we identify important issues for the next generation model-based vision system. Work is underway on the "successor" to ACRONYM.

ACRONYM employs limited image shape descriptions -- all shapes were approximated by trapezoids or ellipses. These shapes are sufficient for the coarse reasoning necessary to constrain the viewpoint but obviously cannot describe all the possible image projections in our set of switch parts. Much current research in computer vision is devoted to the description of shapes in an image.

Many of the parts for the switch assembly contain fine details such as rounded corners and small slots which are important for successful assembly. While it is possible to model these fine details in ACRONYM, the rules which embody the reasoning about them would be complex, awkward and numerous. In the future we hope to use a uniform representation to avoid the case analysis that becomes unmanageable in ACRONYM. Case analysis is acceptable for improved performance in common cases but must be supported by general reasoning.

ACRONYM uses the structure of objects (the relative length of each subpart, the angle between two subparts, etc.) for feature prediction and image interpretation. It is quite competent with objects having skeletal structure such as objects recognizable from stick figure drawings. However, several of the switch parts have similar skeletal structure (e.g., all the washers have the same structure). Additional relationships will be required for classes of objects which lack definitive skeletal structure.

ACRONYM uses graph matching (subgraph isomorphism) to find image features (the interpretation graph) represented in the prediction graph. Relationships connect the shapes for both graphs. In real images, some weak relationships may be lost due to noise or artifacts from edge detection. Subgraph isomorphism

makes no allowance for partial evidence - a node is either completely consistent with all the predicted relationships or it is inconsistent. Intelligent interpretation of image data requires matching groups of features and relationships on the basis of partial evidence.

Conclusion

The original ACRONYM vision system demonstrated the power of geometric reasoning for image feature prediction and image interpretation. It was very successful in recognizing objects from skeletal structure. The modifications for the Intelligent Task Automation project have added a new capability to reason about and recognize objects with holes. A representation for springs was developed and new relations for spring feature prediction were implemented. Results on successful and unsuccessful image interpretations were presented.

Acknowledgements

This work was supported by the Air Force Wright Aeronautical Laboratory under Contract Number F33615-82-K-5105 and Honeywell Inc. under Subcontract Number 512403-FA through the ITA program.

References

1. Honeywell Inc., "Enhanced Enabling Technology Development and Feasibility Demonstration Plan, Intelligent Task Automation Subsystem," April 29, 1983, Air Force Wright Aeronautical Laboratory Contract No. F33615-82-C-5092
2. Brooks, Rodney A., "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intelligence*, Vol. 17, 1981, pp. 285-348, a longer version is available as Stanford AIM-343, Department of Computer Science, Stanford, CA 94305
3. Brooks, Rodney A., "Model-Based Three-Dimensional Interpretations of Two-Dimensional Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, No. 2, March 1983, pp. 140-150.
4. Binford, Thomas O., "Visual Perception by Computer," Invited paper at IEEE Systems Science and Cybernetics Conference, Miami, Dec. 1971
5. Nevatia, Remakant and Binford, Thomas O., "Description and Recognition of Curved Objects," *Artificial Intelligence*, Vol. 8, 1977, pp. 77-98.
6. Lieberman, L., "Model-Driven Vision for Industrial Automation," in *Advances in Digital Image Processing: Theory, Application, Implementation*, Peter Stucki, ed., Plenum Press, 1979.

Iconic to Symbolic Processing
Using a Content Addressable Array Parallel Processor

Daryl Lawton, Steve Levitan, Chip Weems,
Edward Riseman, Allen Hanson

Department of Computer and Information Science
University of Massachusetts, Amherst, MA 01003

ABSTRACT

We discuss the design of a large scale Content Addressable Array Parallel Processor (CAAPP) for low, medium and high level vision processing. This new architecture combines associative processing with global broadcast and response to and from an array of cells, and array processing via local cellular square neighborhood computation. The capabilities of the CAAPP allow us to close the feedback loop between high level processing and low level processing by supporting communication between different representations of an image. The CAAPP would provide a means of mapping the signal level (iconic) pixel-based representation of an image into a symbolic intermediate level representation suitable for high level vision processing.

1.0 INTRODUCTION

The Content Addressable Array Parallel Processor (CAAPP) is a new architecture specially designed for machine vision processing [WEE82, 84a, 84b]. The CAAPP is a "processor per pixel" parallel image architecture which represents the synthesis of both content addressable processors (such as STARAN or ASPRO [BAT82]) and mesh connected parallel array processors (such as ILLIAC IV [BAR68] or CLIP-4 [DUF8]). The full system will augment the CAAPP array via its controller with a host processor such as a VAX/11/780 or a LISP machine. The resulting architecture can be used for both associative and array type operations encountered in image processing and computer vision tasks to produce simple solutions that are difficult for parallel machines which provide only one of these capabilities (Figure 1).

The CAAPP will be capable of performing all low level image processing tasks, but more importantly it will provide a mechanism for transforming low level image data into higher level symbolic data directly --- without mediation (or serial processing) by symbolic "host" processors. Thus, it allows for a new style of high level algorithms where processing decisions can be based on direct global feedback information from the processing elements. We have closed the feedback loop between low level and high level processing by providing a control interface. The key to this capability is the provision of fast global feedback operations from the array to the controller (Figure 2).

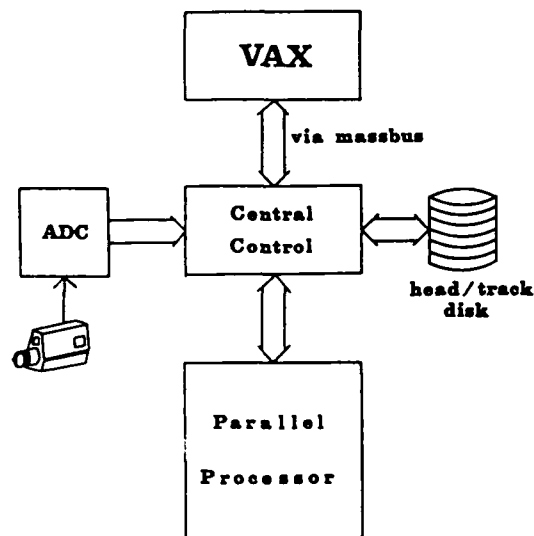


Figure 1

In the next sections we discuss our view of the goals of general machine vision in terms of image interpretation. We continue with a discussion of the characteristics of our design of the CAAPP as a machine for vision processing. We give a description of the CAAPP and show the utility of such a design by discussing some of the algorithms we have examined during our design process. Then we illustrate the process of mapping a low level iconic representation of an image into a symbolic representation suitable for high level processing.

2.0 THE MACHINE VISION PROBLEM

The processing requirements needed to solve the machine vision problem are not well understood. The difficulty is that general machine vision is far from being solved, and is currently a rapidly evolving area of research. At this point in time, no one can give a detailed algorithmic specification for a general vision interpretation system. However, it is possible to give a list of features that must be present in any machine that is to be used to significantly advance the vision problem. We believe that if such machines are built, they will greatly facilitate research and clarify many issues in machine vision development.

By machine vision, or image understanding, we mean much more than image processing which, usually, refers to the enhancement and classification of images. The goal of machine vision is the automatic transformation of an image to a symbolic form that represents a description and an understanding of the content of the image. In general, the machine vision problem subsumes the tasks performed in normal image processing.

The image understanding process can be thought of as an iconic to symbolic (or signal to symbol) transformation. The input image data is essentially an array of signal data and forms an iconic representation of the real world. To perform image interpretation the machine must transform this data into a symbolic form. The transformation is from low level information (e.g., the pixel at coordinates [112,47] has a blue intensity value of 1/) to symbolic representations of objects in the scene, in terms of predefined knowledge about objects in the world (e.g., region 75 in the image is an instance of the object class HOUSE--DOOR). This task involves monocular static image interpretation as well as integrating information from multiple sensory sources including stereo input, motion sequences, and laser ranging information.

From our perspective, the machine vision problem will be described as involving three levels of processing. These are referred to as the low, intermediate and high levels (Figure 3). The low level consists mainly of operations on pixels and local neighborhoods of pixels. This may involve segmentation algorithms to partition pixels into regions of similar color and texture properties,

and to extract lines via intensity and color discontinuities at local edges. The result of what we call low level processing is a transformed image with labeled regions and line segments. However, we are assuming that no operations relating different image events have been performed nor have there been any inferences on the object identity of these events.

The intermediate level of representation provides an interface between the low and high levels of representation, that is, between pixel-based representation and symbolic elements representing visual knowledge stored in a database. In the UMASS VISIONS system [HAN78a,b,HAN83], which is the environment in which most of this research was conducted, the intermediate level consists of a symbolic description of the two dimensional image in terms of regions and line segments (that are still in registration with the raw image data) as well as their associated attributes which can be used in the interpretation process. In some systems this level would consist of representations of surfaces, or more generally, "intrinsic" features of the physical environment [BAR78,MAR82].

Intermediate processing includes several kinds of activities. First is the set of bottom-up tasks which are needed to complete the intermediate level of representation. This includes the extraction of the features for regions, lines, and vertices as well as the relations between these entities. The results of this processing are representations of image entities:

- * Regions have information about intensity, color, texture, location, size, major axis orientation, compactness, labels of bordering segments and adjacent regions, etc.

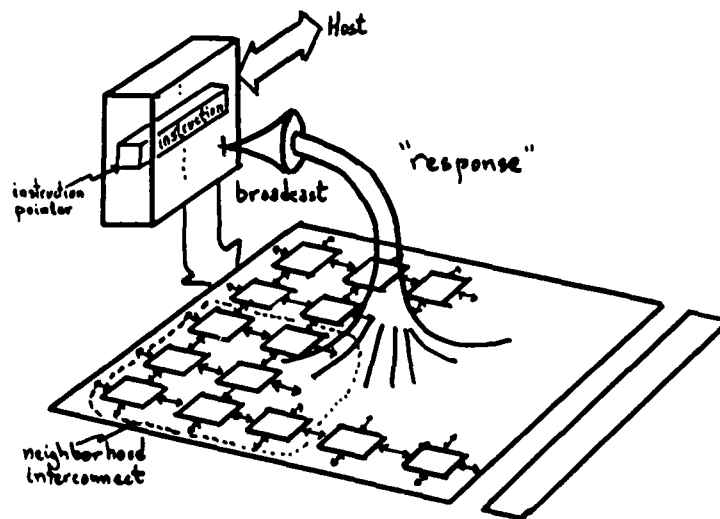
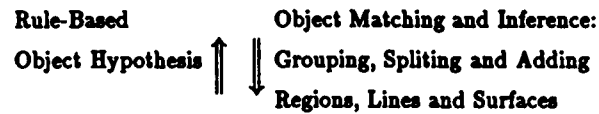


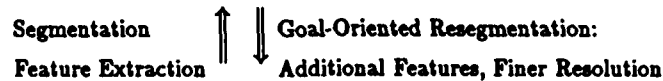
Figure 2

Communications and Control Across Multiple Levels of Representation

High Level - Schema - Symbolic Descriptions of Objects - Control Strategies



Intermediate Level - Symbolic Description of Regions, Lines, Surfaces



Low-Level - Pixels - Arrays of Intensity, RGB, Depth (Static monocular, stereo, motion)

Figure 3

- * Line segments have information about location, orientation, length, width, contrast, labels of adjacent regions, etc.
- * Vertices have information about location, what line segments they connect, their curvature, etc.

These representations are stored along with the normal "pixel" information in the processing elements which hold the respective image objects. Note that the relationships between the objects in the image and objects in the world are not yet elucidated.

The second group of intermediate processing activities involve grouping, splitting, and labeling processes, in either data-directed or knowledge directed modes (i.e., bottom-up or top-down) to form intermediate events which more naturally match stored object descriptions. Some operations in this class are:

- * Labeling points of high curvature on the perimeter of a region.
- * Merging co-linear line segments based on the properties of their adjacent regions.
- * Merging adjacent regions based on their relationship to shared line segments.

The high level processing controls the intermediate level of processing where the symbolic two-dimensional representations of the intermediate level must be related to object descriptions stored in a knowledge base. The object descriptions represent information about the three-dimensional world in a representation that might be used to form matches. Thus, objects will be represented in terms of significant region, line, and surface

features that are expected, spatial relations to other objects which might be identified, and a set of control strategies for partial matches, for merging fragmented regions (possibly due to texture), for filling in missing information (possibly due to occlusion), and finally for inferring the presence of related objects. The result of high level processing is a symbolic representation of the content of a specific image in terms of the general stored knowledge of the object classes and the physical environment.

Information flow between representations is in both directions. In the upward direction, the communication consists of segmentation results from multiple algorithms, and possibly from multiple sensory sources. It also involves the computation of a set of attributes of each extracted image event to be stored in a symbolic representation. The summary information and statistics allow processes at the higher levels to evaluate the success of lower level operations. It is also the mechanism for the passing of actual symbols. In the downward direction the communication consists of commands for selecting subsets of the image, for specifying further processing in particular portions of the image, and requests for additional information in terms of the intermediate representation.

From the above description of the machine vision problem, a set of three general requirements for a machine vision architecture can be deduced. The first is that the machine must be able to load (and possibly dump) a complete image in well less than a 33 millisecond frame time (or in parallel with the actual processing of a previous frame). Loading a 512 x 512 RGB color image in under one frame time represents a rather high data transfer rate.

Second, since a great number of low level operations will be needed to support processing at the higher levels, the speed requirement indicates the necessity of the pixel per element class of mesh connected (local neighborhood) cellular array processors. It is generally recognized that these provide the greatest speed in performing low level image operations.

Most important of the architectural requirements, however, is that a general vision machine should provide mechanisms for communicating information and control both up and down through the three levels of representation. The control program must be able to determine the necessary summary information quickly, so that it can try a variety of processing approaches to produce the best results. This type of communication is necessary to permit the autonomous transformation of an image to a set of meaningful symbols. For this reason, the mechanisms that provide the summary information must be applicable to both pixel and symbol data.

To summarize --- a key issue in achieving an effective architecture is the ability to maintain the low and intermediate representations, pixels and symbolic region, line and surface representations simultaneously in the same machine. The necessity of dumping an image out, for evaluation by a sequential program, must be avoided at all cost. It is too time consuming to transfer the volume of information contained in an image. Even if it took no time to dump the information, the time required for serial evaluation would still be too great. Dumping an image for outside evaluation defeats the entire purpose of having a special parallel processor for computer vision. Instead, the vision machine must be able to provide enough feedback to the controlling processor to allow all of the operations to take place within the vision machine itself.

3.0 AN ARCHITECTURE FOR MACHINE VISION

The CAAPP has been designed to support vision processing research. It is also sufficiently general that new approaches, to the various aspects of vision, can be easily implemented on it. It is quite simple to build special purpose machines that implement particular image processing algorithms with great speed. However, as mentioned above, machine vision research is a dynamic, rapidly changing area. New algorithms are constantly under development and experimentation. A vision machine must therefore be sufficiently fast and general to allow complex experimentation up to the interpretation level.

The basic architectural issues to be addressed for vision stem from the requirements of the problem:

- * The ability to process both pixel and symbol data
- * A fast processing rate
- * The ability to select particular subsets of the pixels for special processing

- * Feedback mechanisms that allow focussing of attention and data-directed processing, without having to dump the image for external evaluation.
- * The ability to transform an image into a set of meaningful symbols that describe it.

The solution that we have developed is a machine that is a fusion of mesh connected cellular array processors with associative or content addressable parallel processing capability. Previous research has shown that a mesh connected cellular array is a structure that is extremely well suited to performing basic image processing tasks. With one processing element per pixel, such a machine can perform many of the basic image processing operations, including both the pixel and local neighborhood classes of operations, very quickly. The problem with the cellular arrays that have been proposed is that they generally do not provide for selective processing of pixel subsets (such as collections of regions or line segments), nor do they supply feedback to the controller. In other words, they do not provide the necessary bidirectional communication between symbolic processing and pixel processing. An image is simply loaded, some operations are applied to it, and then the image is returned for external sequential processing or human presentation.

Research on content addressable parallel processors (CAPPs) has always emphasized selecting and processing arbitrary subsets of the data elements, providing feedback to the controller and doing whatever is necessary to keep from having to move data in and out of the processor. This is because the time required for loading the data, which is roughly equivalent to the time to serially process the data with one operation, must be included in the total processing time. In order to claim any significant speed increase over a serial processor, a CAPP must be able to average the data load time with a large number of parallel operations. One way of achieving this is to reduce the number of times that the data must be transferred in and out, by eliminating the need to externally evaluate the results of processing. This can be done by providing global summary mechanisms that feed back to the controlling processor, thereby allowing it to perform the evaluation of the processing without removing the data from the processor.

3.1 DESCRIPTION OF THE CAAPP

The CAAPP would consist of sixteen thousand processing elements arranged as a 128 x 128 square array. This design is intended to be expandable up to at least a quarter of a million processing elements in a 512 x 512 array. The initial 16K processor parallel machine will have an effective operating speed several hundred to a thousand times that of the fastest sequential processor available today. The CAAPP would be connected via its own controller to a VAX-11/780, LISP machine, or some other general purpose computing machine which would provide both the algorithm development environment and the operating environment for the system.

The machine would be constructed as a square grid of 128 x 128 processing elements (or cells or P.E.s). We intend to extend this prototype to 512 x 512, which corresponds to the usual number of pixels in a digitized image. Each cell contains 128 bits of storage, five register bits, and a one-bit ALU for bit serial arithmetic and logic functions. Information in each cell can be moved North, South, East, or West on the array so that neighboring cells can communicate with each other. The 128 x 128 memory array is controlled by a microprogrammed controller capable of issuing an array command every 100 nanoseconds. If the controller is interfaced to a VAX, it will be able to receive macro instructions from the VAX as fast as the VAX can issue them.

The machine allows global broadcast from the controller to all cells, an activity bit set by each cell for its response, and the global response from the array of cells to the controller in terms of a count or some/none functions. In particular, a comparand may be broadcast from central control and cells whose contents fail to match the broadcast comparand will be turned off so that exact match to comparand, greater than (less than) comparand, maximum, and minimum searches may be performed in parallel on all cells of the memory.

The individual chips we are designing will contain 64 cells in an 8 x 8 array in a 45-pin package (Figure 4, Table 1). Each PC board will contain 64 chips in an 8 x 8 array (64 cells x 64 cells) and the memory as a whole will contain 4 such cards (expandable to 64 in an 8x8 array) giving the overall 128 cell x 128 cell memory design. Through a clever organization of the architecture we have managed to reduce the number of off-card connections to only 146 lines per card (Table 2), thereby eliminating what has been a major source of unreliability in other parallel processors.

64 Processing Element Chip Pin List		64 Chip PC Card Connection List	
Instruction	25	North Communication	16
Communication	8	South Communication	16
Quadrant Enable	4	East Communication	16
Comparand In	1	West Communication	16
Some/None Out	1	Column Select	16
Read/Hold Count	1	Row Select	16
Shift/Wait Count	1	Instruction	25
Clock	2	Comparand In	1
Power/Ground	2	Some/None Out	1
		Read/Hold Count	1
		Shift/Wait Count	1
		Board Count Latch	1
		Board Count	13
		Board Count Select	3
		Clock	2
		Power/Ground	2
Total	45	Total	145

Table 1

Table 2

The chips and the PC cards are designed to be independent of the overall size of the memory array. All array size dependent functions are implemented in a single column and row of "edge cards" that lie conceptually to the left and below the leftmost column and the bottom row of the array. Thus we will be developing not only a large parallel processor, but also a set of building blocks that can be easily assembled to make other special purpose machines tailored for specific applications.

Images are loaded into the CAAPP in a parallel/serial scheme, one scan line at a time. This takes 1.64 milliseconds, or about 1/20th of of a frame time for a 16 bit (color) image. In addition to the array movement operations and the usual content addressable functions, three important global functions are included. These are: (1) report whether any of the cells is a "responder" (has 1 in its X register), (2) count number of "responders", and (3) find the "first" responder. Together these provide the key to adaptive processing techniques. For example, an image enhancement algorithm could adapt automatically to different light levels in different parts of the image in order to extract the same amount of detail from all parts of the image.

Communication Network for 64 Processing Element Chip

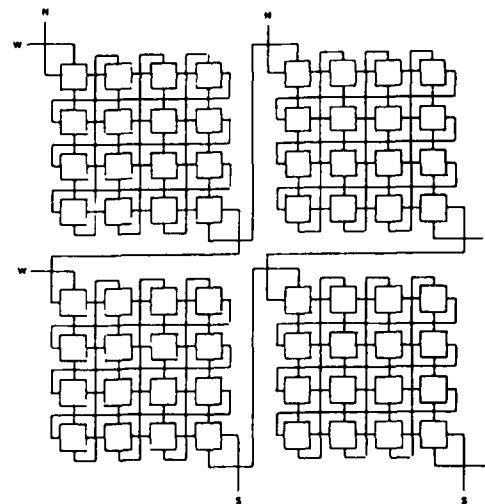


Figure 4

3.2 THE PROCESSING ELEMENTS

Each processing element, or cell (Figure 5) is a bit serial processor consisting of a bit serial ALU, 128 bits of memory (M), local (and global) interconnection hardware, and five single bit registers:

- X The primary accumulator bit, which is also used for communications.
- Y The second accumulator bit.
- Z The carry bit, used for arithmetic operations.
- A The activity bit, used for enabling and disabling this cell on any given operation.
- B The secondary activity bit, used as a temporary storage for activity "flags".

Functional Diagram of One Processing Element

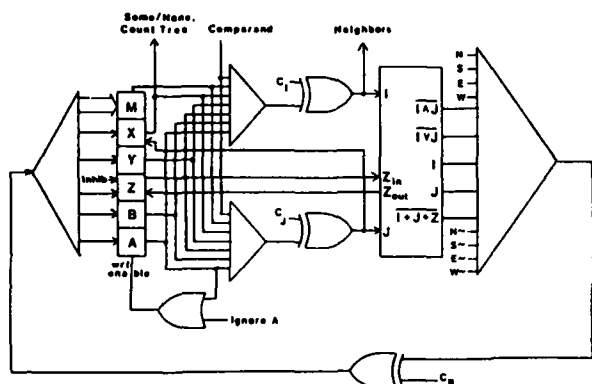


Figure 5

Processing Element Micro-Instruction Set

1	C ₁	Source ₁	C ₂	Source ₂	Function	C ₃	M ₁	Dest.	Address
---	----------------	---------------------	----------------	---------------------	----------	----------------	----------------	-------	---------

1	C ₁ , C ₂ , C ₃	M/R
0	Activity Controlled	0 True
1	Ignore Activity	1 Complement
		0 Register Destination
		1 Memory Destination

Source 1, 2	Destination	Function
0 None	0 None	0 CW~
1 M	1 A	1 I
2 X	2 X	2 J
3 Y	3 Y	3 J
4 Z	4 Z	4 J
5 B	5 B	5 J
6 A	6 A	6 J
7 C	7 A ← J	7 CH~
		8 H
		9 S
		10 E
		11 W
		12 H~
		13 S~
		14 E~
		15 W~

Figure 6

Figure 6 shows the basic micro-operations performed by each cell. Instructions are of the form: "select two sources, perform some function on them, and store the result in some destination." Instructions involving the 128 bit memory must use the same location for read-modify-write operations.

It is interesting to note that this machine has, in a very real sense, sixteen thousand separate parallel processors. As an SIMD machine, it can readily simulate STARAN, ILLIAC-IV, or any of the rectangular array systolic machines proposed by Kung [KUN82]. While there will, of course, be a loss in speed due to simulation, most of this loss is attributable to the bit serial nature of the arithmetic and logical operations. In most cases this is more than compensated for by the greater parallelism of our design.

3.3 IMAGE PROCESSING ALGORITHMS

Table 3 lists 28 algorithms of differing complexity which we have examined during the course of our design of the CAAPP. This is by no means an exhaustive list, however it does show the wide variety of tasks that the CAAPP will support. Table 3 indicates, for each algorithm, whether or not it makes use of intercell communications, the some/none report mechanism, the response count mechanism, broadcast data, and local cellular processing. Of course, all of the algorithms use the instruction broadcast mechanism.

Algorithms Used During CAAPP Evaluation

Algorithm	Micro Coded	Broad-cast Data	Local cell ALU	Inter-cell Comm.	Some None	Fast Count
Basic Macro Operations						
Response count	Yes	No	No	No	No	Yes
Exact match	Yes	Yes	Yes	No	No	No
Greater than	Yes	Yes	Yes	No	No	No
Less than	Yes	Yes	Yes	No	No	No
Select greatest	Yes	No	Yes	No	Yes	No
Select least	Yes	No	Yes	No	Yes	No
Select first	Yes	No	Yes	Yes	Yes	No
Add constant	Yes	Yes	Yes	No	No	No
Subtract constant	Yes	Yes	Yes	No	No	No
Add fields	Yes	No	Yes	No	No	No
Subtract fields	Yes	No	Yes	No	No	No
Multiply constant	Yes	Yes	Yes	No	No	No
Divide constant	Yes	Yes	Yes	No	No	No
Multiply fields	Yes	No	Yes	No	No	No
Divide fields	Yes	No	Yes	No	No	No
Simple Image Processing Operations						
Game of life	Yes	Yes	Yes	Yes	No	No
Gaussian smoothing	Yes	Yes	Yes	Yes	No	No
Sobel edge extract	Yes	Yes	Yes	Yes	No	No
Histogram	Yes	Yes	Yes	No	No	Yes
Rotate model view	Yes	Yes	Yes	Yes	Yes	No
Higher Level Image Processing						
Connected components	Yes	Yes	Yes	Yes	Yes	No
Motion parameters	No	Yes	Yes	Yes	Yes	Yes
Other Application Areas						
Center of mass	Yes	Yes	Yes	Yes	No	Yes
Geocorrection	Yes	Yes	Yes	Yes	Yes	No
Square grid sort	No	No	Yes	Yes	Yes	No
Real time LISP	No	Yes	Yes	No	Yes	No
Neural networks	No	Yes	Yes	Yes	Yes	No
Semantic networks	No	Yes	Yes	Yes	Yes	Yes

Table 3

What Table 3 actually points out is which of the algorithms use purely the associative aspect of the processor, which algorithms use purely the square grid aspect, and which algorithms use both aspects of the processor. Purely associative algorithms are indicated in the Table by a "no" in the Intercell Communications column. For those algorithms with a "yes" in the Intercell Communications column, if the corresponding entry in both the Some/None column and the Fast Count columns is "no", then the algorithm uses purely the square grid aspect of the CAAPP. Those algorithms with a "yes" in the Intercell Communications column and a "yes" in either or both of the Some/None and Fast Count columns use both the associative and the square grid aspects of the CAAPP.

4.0 ICONIC TO SYMBOLIC PROCESSING

A basic step in the functioning of autonomous, general purpose vision systems is the association of symbolic descriptions with the results of segmentation, region, and edge extraction processes. Such a representation acts as a data base which is accessed by recognition and grouping processes to determine the relations between different image structures. The extraction of such a representation has been characterized as the iconic (or image) to symbolic mapping problem and is a critical issue for evaluating proposed architectures for real time image processing.

There are several examples of such spatially tagged, intermediate level symbolic representations: the primal sketch of Marr [MAR82], the curvature primal sketch of Asada and Brady [ASA84], the RSV structure of the VISIONS system [HAN78a,b], the patchy data structure of Ohta [OHT80], Haralick's [LAF82] topographic classification of digital image intensity surfaces, and several others. These representations are organized to be accessed by various processes for the recognition of world objects. Since world knowledge and control strategies in vision systems are expressed in terms of symbolic, relational structures, these representations act as a necessary level of interface to the results of low level image operations.

We have begun investigating iconic to symbolic processing on the CAAPP motivated by its combination of features from both associative processors and square grid array processors as discussed above. This combination gives it the capability of turning raw images into symbolic descriptions in the same memory locations where the segmentations from low level processes are stored. In this section we will present one example of the many possible iconic to symbolic transformations of image data possible in the CAAPP.

The particular representation we have been developing is a version of the RSV representation of the VISIONS image understanding system [HAN78a,b] in which the basic entities are Regions (connected sets of pixels); Segments (portions of the contours surrounding regions); and Vertices (selected points along contours). Each of these

entities has specific attributes (such as area and extent for regions; length and orientation for segments). There are also specific relations between these entities (such as adjacencies between regions and edges). Associating this representation with an image consists of the following stages of processing:

- 1) An image is loaded into the CAAPP. This involves some portion (18 bits assuming a 512x512 image) of the 128 bit memory of each CAAPP cell.
- 2) Each cell has its coordinates in the CAAPP array computed and stored in another 18 bit portion of its 128 bit memory (required for a 512x512 image).
- 3) A segmentation procedure is applied. We have experimented with simple segmentation schemes based upon differences of Gaussian convolutions followed by thresholding to extract zero-crossings [MAR82] and histogram-guided segmentation techniques. Both of these procedures are very rapid in the CAAPP and can be made selectively sensitive to different ranges of contrast and spatial frequency information. The segmentation results are stored in another portion of each CAAPP cell called the region property field. The size of the property field depends on the segmentation technique. For binary regions resulting from a single threshold only 1 bit is required. For general segmentation techniques, the requirements depend on the number of labels associated with ranges of the defining properties.
- 4) Points along the boundaries of regions are determined. This is a local computation over the neighborhood of a cell to determine if the values in its region property field are different than those in surrounding cells. At this stage local edge connectivity and the number of adjacent regions at a point can be determined. The number of adjacent regions at a point is called the local connectivity type. For a square pixel grid, the local connectivity values can range from zero to four.
- 5) An operator for extracting points of significant curvature is applied to the extracted boundary points. This is a local computation over the immediate neighborhood of a boundary point and finds points along a contour of distinctive curvature. These points are tagged as being vertices (the V of RSV) and will be treated as the endpoints of boundary segments.
- 6) The vertices extracted in step 5 are then treated as seeds in a contour message passing process. The message passing involves moving the coordinates of each vertex along the boundaries that intersect with it. This propagation continues along a boundary until another interesting point is encountered. Collisions of these vertex labels at the mid-points of segments are also tagged. Since all the message passing is occurring in unison, the number of steps is maintained as a global count and broadcast to the particular cells at which collisions have occurred. This is a measure of contour length.

7) After stage 6, each vertex point contains its coordinates and those of adjacent vertices to which it is connected along some boundary. Additionally, each tagged midpoint contains the coordinates of the endpoints of its associated segment. The following things are then computed in parallel from these coordinate values at the vertices and midpoints: the distance between the endpoints, the slope of the line, the deviation of the midpoint from the line determined by the two endpoints, and the difference between the number of steps and the summed absolute difference of the components of the segment endpoints (these correspond to measures of the goodness of the linear fit). These values are stored in unused cells near the extracted vertices. These cells are tagged as being either midpoint or segment cells (the S in RSV).

8) Independent of the boundary processing in steps 5-7, a diffusion process is also applied over the values in the region property fields to determine connected components. The component labels are determined from the coordinates of the cells in the region. Collisions between adjacent cells having the same values in the region property field are resolved by letting the one with the least row,col coordinate be the dominating label in the region label field. The particular region cell having the least row, col component is called the Region Cell and is the CAAPP location where information about the corresponding region is stored (The R in RSV).

9) We can then step sequentially through the extracted image structures using the Find First Responder operation of the CAAPP. This operation selects one of the cells that is currently responding to an associative query, turning off all of the other responders. This allows processing to take place on that single cell without affecting any of the other responders. The cell that is selected is the one that would first be encountered if the CAAPP array were to be scanned in normal raster order. For example, we can proceed sequentially through each of the region cells, extract the label of the corresponding region, broadcast it, find the responding region cells, and then compute simple region properties such as area, perimeter length, minimum bounding rectangle, and so forth for the corresponding region. These region values are then stored at (or near) the corresponding region cell. They can also be broadcast and then stored in locations associated with other image structures. This distributes the information and makes parallel processing of relational queries possible. This can be done for each region or only regions having certain properties such as particular size, shape, or image position.

This processing results in attribute lists for particular regions, segments, vertices, and labels associated with interior region and boundary points. The global broadcast and associative properties of the CAAPP then allow us to make queries to select particular image structures, to tag them, and then determine relations between them.

We now discuss the details of the association of these symbolic labels with the raw image in some detail and also discuss the relational processing that can occur with respect to them in the CAAPP.

4.1 LOCAL EDGE AND VERTEX REPRESENTATION

Images are composed of pixels which are separated by edge-links that meet at a point. Associated with each CAAPP processor are the values at the corresponding pixel in the image, the point, and the four edge links incident with the point (Figure 7). The redundant storage of local edge link connectivity simplifies message passing along contours.

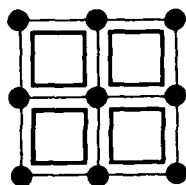
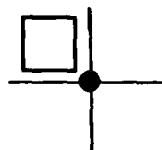


Figure 7



The 128 bits in each CAAPP cell are initially decomposed into fields indicated in Table 4. Each processor has its coordinates in the image array stored with it. The coordinate grid is created in the CAAPP processing array by a series of broadcasts from the controller using different patterns of row and column enable signals, followed by a series of response shifts using the on-chip communication network. Loading the complete 512 by 512 coordinate grid takes 12.7 microseconds. The storage of these coordinates is necessary for computing geometric relations among sets of selected pixels and is a source of unique labels for connected-components analysis by having the label associated with a region be the coordinates of its upper-most, left-most pixel. Note that the required number of bits for this attribute are a function of image size. The raw image values correspond to the actual image which is being operated upon. Again, fewer bits than 18 can be used, or the raw image can be removed from storage once a segmentation has been obtained and processing is restricted to it. The property field corresponds to the value upon which the segmentation is performed, such as the response to some texture measure or the domain of some histogram. The region label field is used for the connected components analysis and must be of the same dimension as the coordinates of the image.

FIELD		BITS
pixel location	(row,col coordinates)	18
raw image value	(6 bit R,G,B)	18
region property		18
region label		18
local connectivity		4
R, S, V or Midpoint		2
working area		50

Table 4

The working area is used in several different ways. For example, our implementation of contour message passing requires 44 bits (8 for storing local edge connectivity traversals and 36 for two sets of pixel coordinates). The interest operator, depending upon resolution and the number of iterations, can require from 24 to 48 bits. Several of the queries require multiple tag bits to be set in particular combinations. Note that all these fields are not needed simultaneously. Many are dependent upon the segmentation being performed and need not be used for anything until the segmentation has been obtained. The contour message passing occurs after the interesting points have been extracted. However it is obvious that larger image memory is needed in more complex processing, if memory swapping between host and the CAAPP array is to be avoided.

4.2 SEGMENTATION

As mentioned previously, we have explored two types of segmentation procedures on the CAAPP: Histogram Guided Segmentation and Zero-crossing extraction using DOGs (Difference of Gaussians).

Histogram based segmentation [OHL75, OHL78, NAG79, NAG82, KOH83, REY84] on the CAAPP involves forming a histogram, determining what region labels to associate with the different ranges in the histogram based upon its peaks and valleys, and then broadcasting the determined labels for the particular ranges of pixel values. Forming the histogram uses the Select Less Than and Count Responders micro sub-routines to select ranges of values (buckets) starting with the lowest range and working up to the highest range. For example, if the range of each bucket is taken to be the maximum range divided by the number of buckets, then the time for the algorithm is dominated by the time to perform the count responders operation for each bucket. The time is: (Number of Buckets) * 1.6 + 0.9 microseconds; A 256 bucket 8-bit histogram would take 410.5 microseconds. The histogram is collected in the central control which collects the counts for each bucket. The resulting histogram is a global histogram for the whole image. Labeling the histogram takes place off the CAAPP in the host processor.

Experience has shown that global histograms are ineffective in images with complex information. We have not explored all the requirements for computing localized histograms and their subsequent remerging in the CAAPP, nor the inherently sequential process of recursive resegmentation. Both of these approaches should be significantly enhanced by incorporating an array of localized controllers in the CAAPP design.

For the initial experiments described here, we have used the segmentations which result from thresholding the difference of Gaussians. Such segmentations are binary images and allow for simple storage of boundaries since there are only two types of regions (greater or less than the zero threshold). For such binary segmentations, there are no type 1 or 3 connectivities and those of type 4 can be removed by removing region labels around type 4 connectivity cells. Thus, it is possible to have single boundaries with only type 2 connectivities. The modifications for other types of segmentations require redundant storage for boundaries of adjacent regions. This involves no new processes in our procedures, but does require the allocation of additional storage in CAAPP memory cells so that the boundary of each region is uniquely represented. A simple technique for this is to remove the outermost pixels from each region to give each region a unique boundary of type 2 connectivity only. The type 2,3, and 4 vertices between these boundaries are then used to store region adjacency information. This does, however, destroy regions of single pixel widths.

There are three basic steps in Zero-crossing extraction using a difference of Gaussians:

- 1) Convolve with Gaussians of different widths and store results separately
- 2) Subtract Results
- 3) Threshold at zero to yield contour and binary regions

This is performed on the CAAPP by taking advantage of the fact that for Gaussians and other smoothing operations, convolutions with large masks are equivalent to multiple convolutions with smaller masks. Thus, processing involves convolving an image with a mask some number of times and storing the result, and then continuing the convolution further and subtracting the result from the one previously stored. Among the masks that could be used are those in Table 5.

On the CAAPP convolution is a "macro" level operation programmed in terms of the basic "micro" CAAPP operations. A discrete, two dimensional, convolution is based on a mask of multipliers that each cell applies to its local neighborhood, forming the sum of the pairwise products of the cell's neighbors with their corresponding mask values. Typically the sum is then scaled in some

Uniform Weighting			Burt's Kernel				
1	1	1	.0025	.0125	.02	.0125	.0025
1	1	1	.0125	.0625	.1	.0625	.0125
1	1	1	.02	.1	.16	.1	.02
			.0125	.0625	.1	.0625	.0125
			.0025	.0125	.02	.0125	.0025

Table 5

manner, and the resulting value is used to update the value in the cell. The algorithm for the convolution can be described as the actions of a single cell with the understanding that each action is performed simultaneously by all of the cells.

For the CAAPP each cell distributes its own data to every cell in the neighborhood. Because every other cell is also doing this, the end result is that the central cell (and hence all cells) gets the data it needs from all of the cells in the neighborhood. The data distribution path is a rectangular spiral out from the center cell. It should be noted that the time required to perform a convolution using the CAAPP is independent of the size of the image (assuming the image is no larger than the array) and only dependent upon the area of the convolution mask. Since the CAAPP does cell-level arithmetic bit-serially, the size of the data values also affects the speed of the algorithm.

A worst case estimate of the time required for such convolutions can be obtained from the formula:

$$T = P(0.8 * N + 0.2 * M + 0.1) + 0.3 * M * (N^2 * P + N + 1)$$

where T is the time in microseconds, N is the number of bits in a pixel value, M is the number of bits in a mask value and P is the number of pixels in the mask area. Under normal circumstances T will be about half of the value obtained from the formula. For 8 bit pixel values, this would give times of:

Mask Size	Time (milliseconds)
3 x 3	0.7
5 x 5	2.1
7 x 7	4.0
11 x 11	9.9

As an example, the binary segmentation resulting from the difference between the image in Figure 8 with the smoothed image derived from it after eight successive convolutions with Burt's kernel [BUR82] is shown in Figure 9.



Figure 8



Figure 9

4.3 LOCAL EDGE AND VERTEX PROCESSING

The next stage of processing determines which pixels are adjacent to boundaries and what the local edge connectivity is. This operation is based upon the simple comparisons in the following four steps applied to all cells (described with respect to local connectivity of cell A in Figure 10);

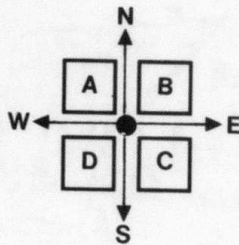


Figure 10

```

if (region_property_field_at_A.ne.
    region_property_field_at_B) =>
    Edge_bit_tag(North) of A = 1 else 0

if (region_property_field_at_A.ne.
    region_property_field_at_D) =>
    Edge_bit_tag(West) of A = 1 else 0

Edge_bit_tag(East) of A =
    Edge_bit_tag(West) of B

Edge_bit_tag(South) of A =
    Edge_bit_tag(North) of D

```

The local connectivity type is the number of set edge bit tags.

4.4 INTERESTING POINT EXTRACTION

The next processing stage extracts "interesting" points along a boundary. These are generally points of high or changing curvature. The procedure we utilize on the CAAPP is tunable for different ranges of curvature. It also involves only local operations over a boundary point and its immediate neighbors to allow for very rapid processing.

Processing begins by associating an orientation vector with each vertex along the boundary using the rules indicated in Figure 11 at the four different boundary orientations possible on a square grid. Once an orientation vector has been associated with each point along the boundary, each point re-expresses its orientation vector by averaging the row and column components of its vector with its immediate neighbors along boundary. The number of iterations of this averaging process corresponds to weighted evaluation of curvature over different neighborhood sizes along the boundary.

Interesting points are extracted where significant changes and variations in orientation occur. This is measured by the sum of the absolute differences between the row and column components of the orientation vectors at the two neighboring boundary points. The interesting points are the local maxima in this measure which also exceed some small threshold.

The determination of the orientation variance measure can also be performed with respect to a particular direction along a curve. In this case it is necessary to assign an orientation (clockwise or counter-clockwise) to points along the boundary of the region. One way to do this is to select some point, choose a direction, and walking around the curve until the point is encountered again. This is a serial and time consuming operation. It is possible to determine boundary orientation through a completely local procedure since each point knows its direction relative to the interior of the region and the global North, East, West, South directions on the CAAPP. To do this, each initial orientation vector at a boundary point samples the edge link connectivity moving in the N,E,S,W directions (clockwise) until a set one is encountered. This assigns a unique neighbor to each boundary point and thus orients the curve (see Figure 12). This direction can be stored to simplify further message passing and walking along a contour. Additionally, given an orientated curve, the direction of curvature can be determined by projecting the difference vector between adjacent orientation vectors onto the direction of boundary orientation.

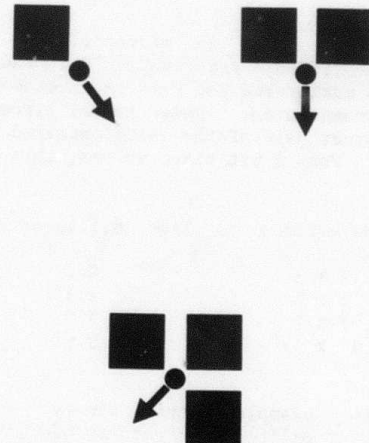


Figure 11

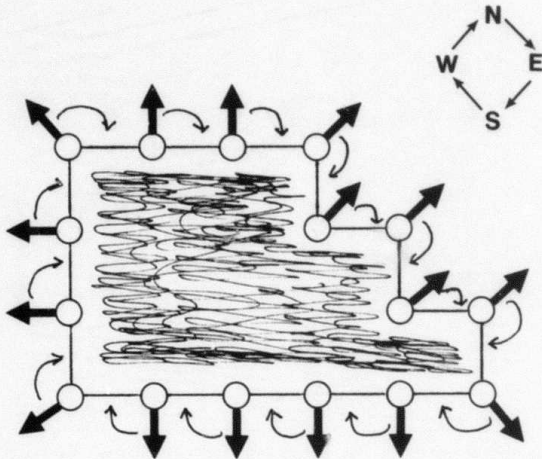


Figure 12

Figures 13a,b,c show the positions of the interesting points with respect to the contours from figure 9 after 25, 50, and 125 iterations of the averaging procedure. The average orientation vector length rapidly decreases for very small regions. Figures 14a,b,c show the linear segments between the extracted points for the corresponding number of iterations.

Figure 15a,b,c shows the positions of the interesting points from the contour segments along what is basically the house roof for the different iterations. Figures 16a,b,c show the successive values of the measure of difference between the orientation vectors neighborhoods along this contour and Figures 17a,b,c show the linear segments between these points. The interpolated smoothed curves derived from the orientation vectors are shown in Figures 18a,b,c.

4.5 SEGMENT EXTRACTION VIA MESSAGE PASSING

Segments are then extracted between interesting points. This is performed by having each vertex send out its coordinates along its incident boundaries. These propagate until another interesting point is encountered. The number of steps is maintained by a global step count for a measure of contour length. Finally, when each interesting point has been collided with an appropriate number of time (as determined by its local connectivity), attributes such as slope, distance, and linear deviation are computed in parallel for each line segment. For endpoints, linear deviation is the difference between the number of steps between interesting points and the sum of the component-wise absolute valued difference of the endpoints.

Messages between interesting points are passed in parallel for all points along the perimeter of each region.

1) The operation begins by copying the coordinates of each interesting point into two (redundant to start with) fields in the processing elements (P.E.) which hold such points. We call these message-1 and message-2. We call the coordinate data being sent, messages. Those P.E.'s also set flags in a bit vector (D-Out) representing the four possible directions that the point could have arcs on. There must be exactly two such arcs. This is true at all points, not just the "interesting" ones, since we have already eliminated all but type 2 vertices.

2) For all P.E.'s with Any D-Out bits set we send out single bit flags to the two neighbors on the arcs adjacent to them. This operation is done in parallel, but once each for the four directions with a test for each of four possible D-Out bits.

3) For all P.E.'s on a edge, we check to see if we have bits set on any of our neighbors adjacent to us, on arcs which define the edge of our region. If we do we set the corresponding (D-In) flags. Again every P.E. will "look" in each of the four directions. There will be none, one, or two bits set. If two D-In bits are set, we have a "collision" which means this vertex is at the midpoint of two interesting points. We record that fact in a separate flag in the P. E. memory.

4) For each of the "n" bits which represent the messages being sent, and for each of the four directions (North, South, East, West), we perform the following operations, indexed by "i":

- If the D-Out bit for this direction is set, and there is only one D-out bit set, pick up the ith bit of the message from memory field message-1 and send it out. If this is the second of two D-Out bits set then use the message-2 field.
- If the D-In bit is set for the corresponding direction (South for North etc.) and there is only one D-In bit set, pick up the ith bit of the message and store it the memory field message-1 of the P. E. If this is the second D-In bit set store the message bit in message-2.

5) After all "n" bits of the current messages have been sent, we map D-In bits to D-Out bits such that messages that came in from the North go out to the South. We do this for all vertices that are not interesting points. Messages will stop traveling at interesting points.

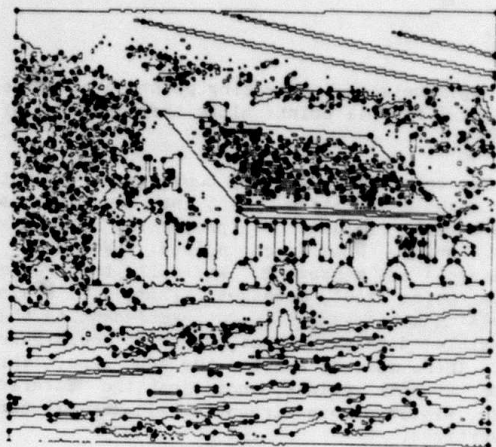


Figure 13a



Figure 14a

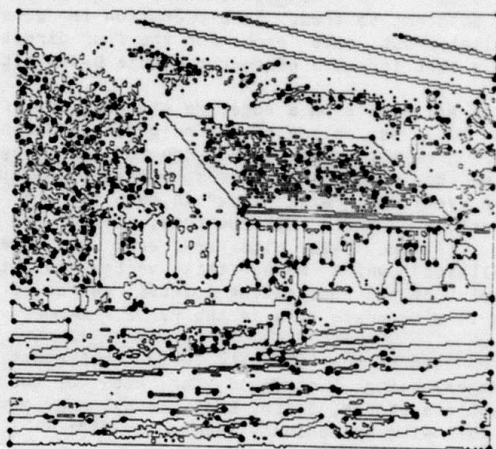


Figure 13b



Figure 14b

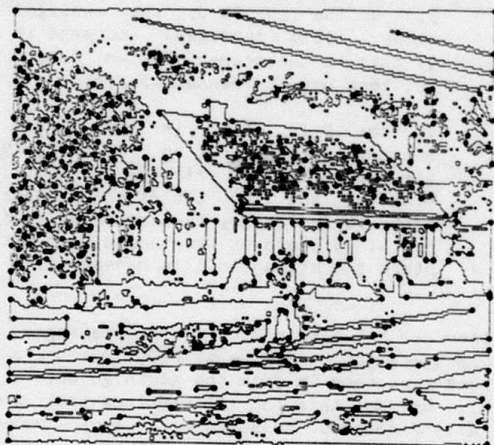


Figure 13c



Figure 14c

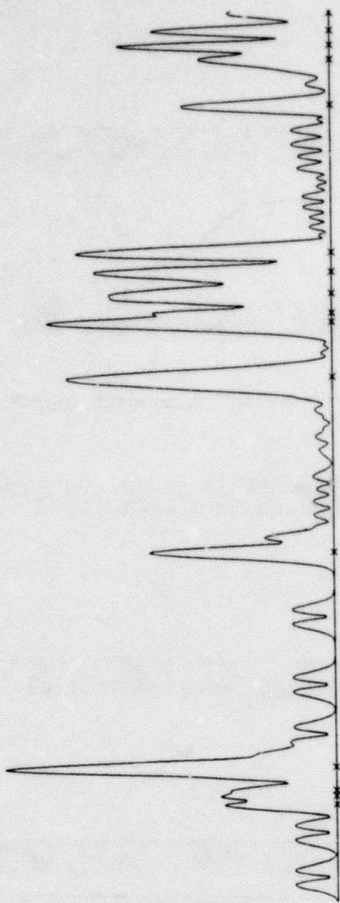


Figure 16a

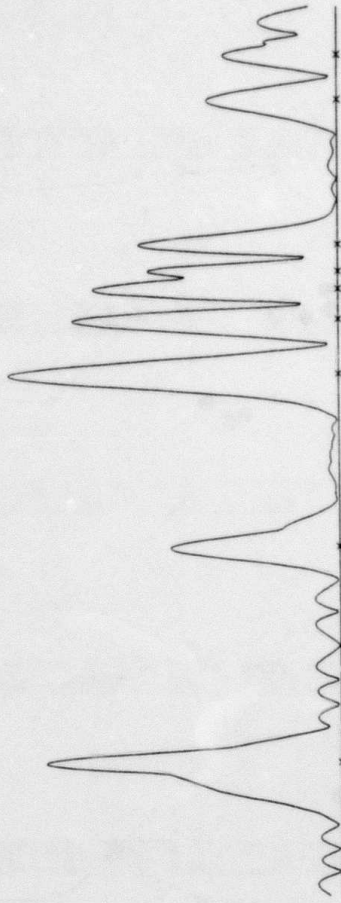


Figure 16b

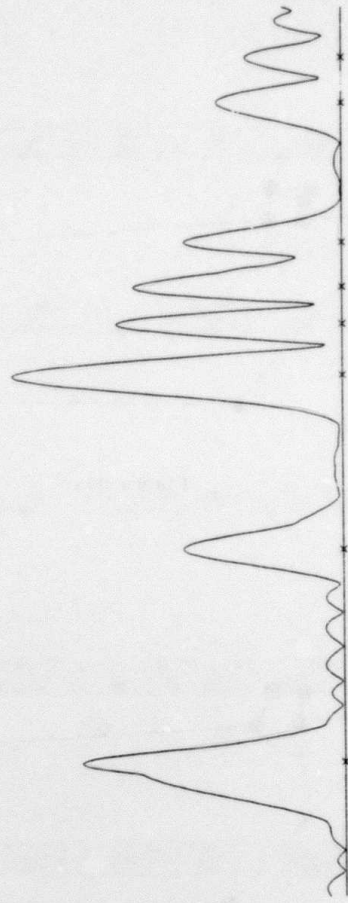


Figure 16c

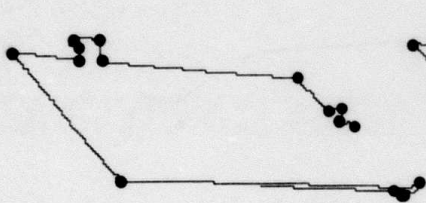


Figure 15a

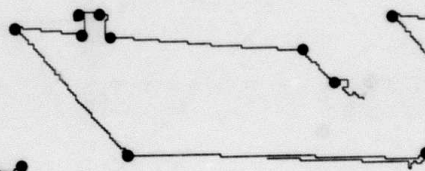


Figure 15b

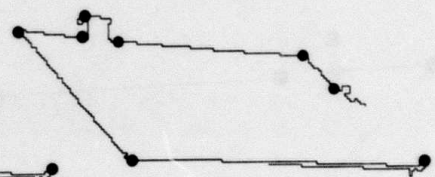


Figure 15c

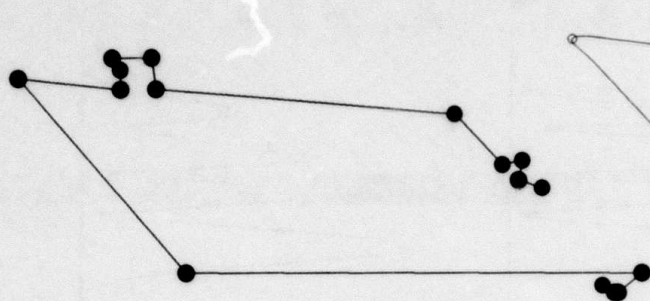


Figure 17a

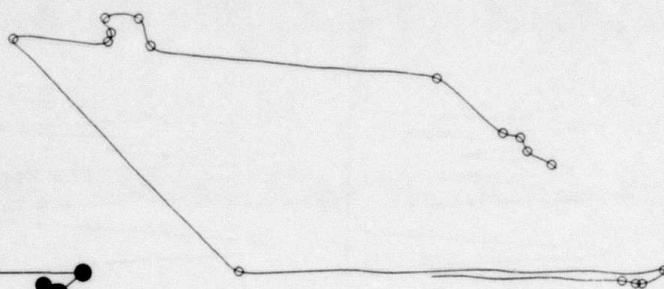


Figure 18a

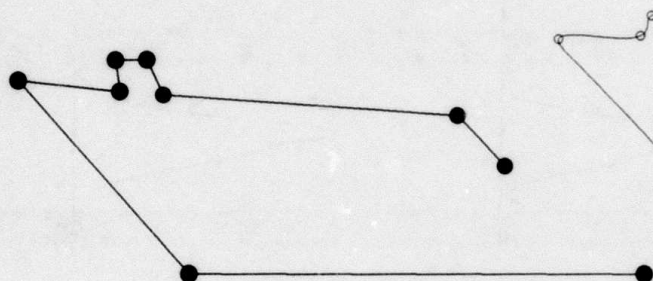


Figure 17b

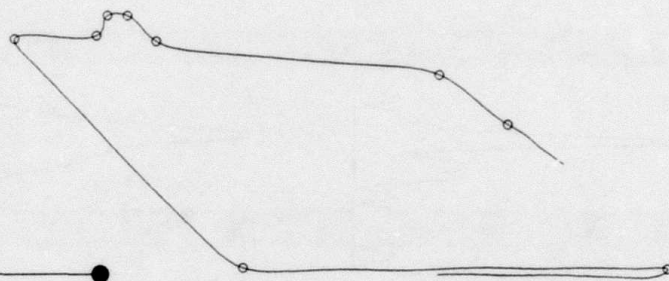


Figure 18b

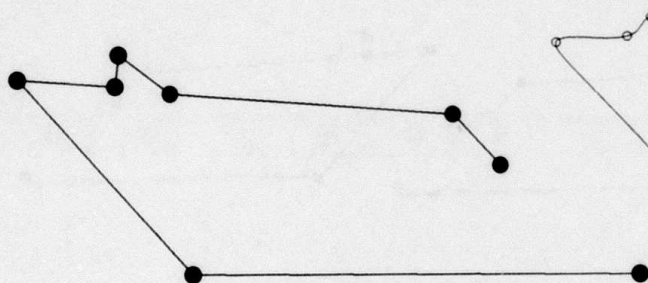


Figure 17c

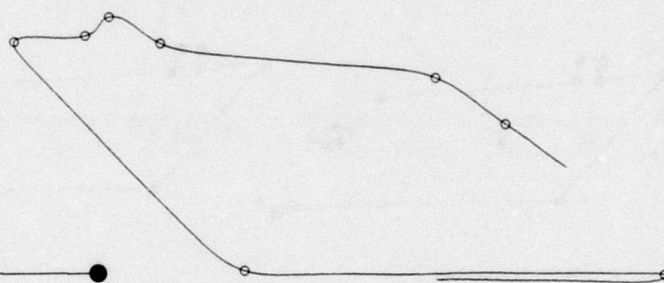


Figure 18c

6) If there are no vertices with D-Out bits set then all messages have reached the end of their travel and the algorithm terminates.

4.5 REGION EXTRACTION

Region extraction determines connected sets of pixels with identical values in their segmentation property field and particular spatial properties of these connected sets (area, perimeter length, minimum bounding rectangle). The connected components operation is done by locally propagating labels in parallel from points in regions. The labels we use are the processor coordinates of the region points with conflicts resolved by letting the label corresponding to the least row, col component dominating. This operation would take about 10 milliseconds for all regions less than 128 pixels in diameter, with the processing occurring simultaneously over all regions [WEE84].

4.6 QUERIES

Once a symbolic intermediate level representation has been formed, high level processing can be initiated. For example, object hypothesis can be formed via rule-oriented techniques [WEY83,84], [RIS84]. This could involve broadcasting, sequentially, the range of a set of expected feature values for a particular object and setting weighted responses for regions which match these feature values. While the broadcasting of object features proceeds sequentially, each will take place very quickly so that the strongest responders for several objects can be found in a single frame time.

Some simple examples of these queries could be that large, blue, regions in the upper portion of the image with low texture might be a sky segment. If candidate regions were located with a reasonable degree of confidence, we might hypothesize that the region below sky with long straight edges is a roof; while green segments with high texture of random line orientation could be foliage. Of course, these simple queries should not be viewed as a solution to the interpretation problem. Rather, these types of operations are available to a high level knowledge-based and rule-oriented control structure as a focus-of-attention mechanism to allow it to quickly form object hypotheses which can then be processed more carefully. Spatial relations between objects can be tested and particular regions can be examined in more detail during verification.

A basic type of query is to determine a set of lines having some attribute in common. Once extracted, these can be used to initialize and then constrain an interpretation. For example, the set of long straight lines of some range of contrast are significant, especially in interpreting images of man-made objects such. A sequence of CAAPP operations to do this is:

1) Apply a simple contrast measure at points along a region boundary, such as the absolute value difference between image pixels on opposite sides of the region boundary.

2) Threshold this measure to remove low contrast. Tag these positions

3) Assuming that the contour has been oriented (see section 4.4) and the appropriate directions stored at each boundary position, walk from each vertex point the oriented direction keeping count of the number of tagged bits set in step 2 that are encountered. Incrementing the count is stopped and stored when another interesting point is encountered.

4) The contrast along the segment is a ratio of the number of tagged cells encountered and the segment length.

For example, Figure 19a shows the linear segments which have exceeded a minimal contrast threshold. Figure 19b shows a further threshold on contour length.

Another important task is combining region and line elements in the intermediate representation into particular geometric Figures. This can be done by actually broadcasting the Figure and checking segment attributes in the corresponding areas. A simple example of this is for merging straight lines. Initially, some line segment is selected based upon attribute such as it's length, position, contrast, or the current state of the



Figure 19a



Figure 19b

determination of image relationships. From this, the equation of the line is extracted and the cells in the CAAPP which are within some distance of this line are tagged. This is essentially masking an area in the CAAPP for further processing and involves tagging those set of processor which have coordinates satisfying a set of broadcast geometrical relations. Next, any line segments in this area are tagged. Of those which are extracted, the ones with the correct orientation relative to the initial extracted line are found. These may then be linked together, conditional on other attributes (similar region adjacencies or contrast). Simple shape analysis can also proceed sequentially once a line set as been extracted and a particular one selected. The area surrounding its endpoints is tagged and the occurrence of any other lines from the line set in these areas tagged, and whether it is in a geometrical relation consistent with a particular object. If so, the processing continues, otherwise another segment is selected.

A basic question is when to do things in parallel in the CAAPP via local message passing and when to use the global broadcast mechanism to distribute information and focus processing on particular image structures. The answer to this question will ultimately depend on the form high-level vision systems take. It is generally known that visual interpretation is difficult unless a context can be quickly established, perhaps by heuristic strategies (as in the use of schemas in [WEY84]). This implies that much of the processing is highly focused and prediction-driven. This would mean highly directed queries to the CAAPP after some initial, rough and global operations to establish a context. Additionally, the need for segmentations at different levels of detail and over multiple attributes is obvious. For example, the line representation used by Burns [BUR84] and the various segmentation procedures we have developed can be consistently represented in the RSV structure and the effects of combining them is only beginning to be explored.

5.0 CONCLUSIONS

The architecture presented here is novel, buildable, and opens up a whole range of very high speed computation at the more difficult levels of machine vision. Both characteristics of the CAAPP, array processing and associative processing, seem to be fundamental to parallel vision algorithms. These capabilities allow communication which supports both global and local processing, which in turn supports transformations between representations of the image in a highly flexible manner.

ACKNOWLEDGEMENTS

We would like to acknowledge the pioneering work of Caxton Foster in laying a foundation for our efforts and the help and insights of Michael Callahan.

REFERENCES

- [ASA84] Asada, H. and Brady, M.; "The Curvature Primal Sketch", IEEE Workshop on Computer Vision: Representation and Control, April 30 - May 2, 1984.
- [BAR68] G. H. Barnes, et al; "The ILLIAC IV Computer" IEEE Transactions on Computers Vol. C-17, No. 8; August 1968; pp. 746--757.
- [BAR78] H. Barrow and J. Tenenbaum; "Recovering Intrinsic Scene Characteristics from Images"; in Computer Vision Systems A. Hanson and E. Riseman, eds.; Academic Press; pp. 3-26; 1978.
- [BAT82] K.E. Batcher; "Bit Serial Parallel Processing Systems" IEEE Transactions on Computers Vol. C-31, No. 5; May 1982; pp. 377-384.
- [BUR82] Burt, P.J.; "Pyramid-Based Extraction of Local Image Features with Applications to Motion and Texture Analysis", SPIE Conf. on Robotics and Industrial Inspection, San Diego, 1982.
- [BUR84] Burns, J.B., Hanson, A.R., and Riseman, E.M., "Extracting Straight Lines", pp.482-485, vol 1, 7th International Conference on Pattern Recognition, Montreal, Canada, July 30 - August 2, 1982.
- [DUF78] M.J.B. Duff; "Review of the CLIP Image Processing System"; Proceedings of the National Computer Conference AFIPS; 1978; pp. 1055-1060.
- [HAN78a] A.R. Hanson and E.M. Riseman (eds.) Computer Vision Systems Academic Press; New York; 1978.
- [HAN78b] A.R. Hanson and E.M. Riseman; "VISIONS: A Computer System for Interpreting Scenes" in Computer Vision Systems A. Hanson and E. Riseman, eds.; Academic Press; pp. 303-333; 1978.

- [HAN80] A.R. Hanson and E.M. Riseman; "Processing Cones: A Computational Structure of Image Analysis"; Structured Computer Vision S. Tanimoto, ed.; Academic Press; New York; 1980; Also COINS Technical Report 81-38, University of Massachusetts, December 1981.
- [HAN83] A.R. Hanson and E.M. Riseman; "A Summary of Image Understanding Research at the University of Massachusetts"; COINS Technical Report 83-J5; University of Massachusetts at Amherst; October 1983.
- [KOH83] R.R. Kohler; Integrating Non-Semantic Knowledge into Image Segmentation Processes; Ph.D. Dissertation and COINS Technical Report 84-04; University of Massachusetts at Amherst; September 1983.
- [KUN82] H.T. Kung; "Why Systolic Architectures?" IEEE Computer; January 1982; pp. 37-46.
- [LAF82] T.J. Laffey, R.M. Haralick, L.T. Watson; "Topographic Classification of Digital Image Intensity Surfaces", IEEE Workshop on Computer Vision: Representation and Control, August 23-25, 1982.
- [LAW84] D.T. Lawton; Processing Dynamic Image Sequences from a Moving Sensor; Ph.D. Dissertation and COINS Technical Report 84-05; University of Massachusetts at Amherst; February 1984.
- [MAR82] D. Marr; Vision; W.H. Freeman; San Francisco; 1982.
- [NAG78] M. Nagao and T. Matsuyama; A Structural Analysis of Complex Aerial Photographs Plenum Press; New York; 1980.
- [NAG79] P.A. Nagin; Studies in Image Segmentation Algorithms Based on Histogram Clustering and Relaxation Ph.D. Dissertation and COINS Technical Report 79-15; University of Massachusetts at Amherst; September 1979.
- [NAG82] P.A. Nagin, A.R. Hanson and E.M. Riseman; "Studies in Global and Local Histogram-Guided Relaxation Algorithms" IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI-4); May 1982; pp. 263-277.
- [OHL75] R. Ohlander; Analysis of Natural Scenes Ph.D. Dissertation; Carnegie-Mellon University; April 1975.
- [OHL78] R. Ohlander, K. Price, and R. Reddy; "Picture Segmentation Using a Recursive Region Splitting Method" Computer Graphics and Image Processing Vol. 8; 1978; pp. 313-333.
- [OHT80] Ohta, Y.; "A Region-Oriented Image-Analysis System by Computer", Ph.D. Thesis, Kyoto University, Department of Information Science, Kyoto, Japan, 1980.
- [PAR80] C.C. Parma, A.R. Hanson and E.M. Riseman; "Experiments in Schema-Driven Interpretation of a Natural Scene"; COINS Technical Report 80-10; University of Massachusetts at Amherst; April 1980.
- [REY84] G. Reynolds, N. Irwin, A. Hanson and E. Riseman; "Hierarchical Knowledge-Directed Object Extraction Using a Combined Region and Line Representation" Proceedings of the Workshop on Computer Vision: Representation and Control; April 30-May 2, 1984; Annapolis, Maryland; pp. 238-247.
- [RIS74] E.M. Riseman and A.R. Hanson; "The Design of a Semantically Directed Vision Processor"; COINS Technical Report 74C-1; University of Massachusetts at Amherst; January 1974.
- [RIS84] E.M. Riseman and A.R. Hanson; "A Methodology for the Development of General Knowledge-Based Vision Systems"; (to appear) Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems; Denver, Colorado; 3-4 December 1984.
- [STE83] M. Steenstrup, D. Lawton and C. Weems; "Determination of the Rotational and Translational Components of a Flow Field Using a Content Addressable Parallel Processor" Proceedings of the International Conference on Parallel Processing; August 1983; pp. 470-495.
- [WEE82] C. Weems, S. Levitan and C. Foster; "Titanic: A VLSI-Based Content Addressable Parallel Array Processor" Proc. of IEEE International Conference on Circuits and Computers; September 1982; pp. 236-239.
- [WEE84a] C. Weems; Image Processing with a Content Addressable Array Parallel Processor; Ph.D. Dissertation Computer and Information Science Department, University of Massachusetts at Amherst; 1984.
- [WEE84b] C. Weems, S. Levitan, C. Foster, E. Riseman, D. Lawton, A. Hanson; "Development and Construction of A Content Addressable Array Parallel Processor for Knowledge-Based Image Interpretation"; (to appear) proceedings of the Workshop on Algorithm-Guided Parallel Architecture For Automatic Target Recognition; July 1984; Leesburg, VA.
- [WEY83] T.E. Weymouth, J. Griffith, A.R. Hanson and E.M. Riseman; "Rule Based Strategies for Image Interpretation" Proceedings of AAAI-83; August 1983.
- [WEY84] T.E. Weymouth; Using Object Descriptions in a Schema Network for Machine Vision; Ph.D. Dissertation; Computer and Information Science Department, University of Massachusetts at Amherst; 1984.

This research was supported by DARPA under grant N00014-82-K-0464.

SURVEY OF ARRAY PROCESSORS

Hong Seh Lim & Thomas O. Binford

Artificial Intelligence Laboratory, Computer Science Department,
Stanford University, Stanford, CA 94305, USA.

ABSTRACT

A preliminary survey was conducted on commercially available array processors. Array processors from each major manufacturer were chosen for comparison. The computation powers of these machines range from one mega floating point operation per second (Mflop) to a hundred Mflops. Prices range from a few thousand dollars for a single board device to a quarter million dollars for a 100 Mflops machine. A comparison of price, performance, hardware architecture, software availability and input/output interface are summarized. Users' comments on most of the machines are included.

INTRODUCTION

High performance array processors are designed to attach to a general purpose host computer. The combination of a host computer and an array processor can sometimes provide cost effective scientific computations. For a minicomputer or a superminicomputer host, processing speeds common only in the world of Cray and Cyber machines can sometimes be achieved. Although this performance is restricted to repetitive arithmetic functions, the array processors come at prices much less than those of general purpose computers.

In evaluating array processors, two considerations are important. First, quoted performance is usually exaggerated and achievable only for special computations, typically the fast fourier transform (FFT). Quoted operations are not multiply adds, but multiplies plus adds. For the FFT, there are two adds per multiply. Thus 15 Mflops means 5 million multiplies for a machine targeted for the FFT, with one multiplier and two adders. Such a machine would reach only 10 Mflops for convolution. Second, programming array processors requires typically one or two orders of magnitude more time than general purpose computers

because of lack of software development tools and because of hardware particularities of array processors. Software environments are a key issue in selection. We hope to provide more guidance based on our experience with programming one or more machines in later versions of this document.

We have not considered 64 bit floating point operations. They are important for many applications but not for vision, in our opinion.

Mainframes and microcomputers are general purpose machines designed to perform general purpose tasks. Array processors, on the other hand, are designed to handle massive, complex and repetitive calculations. The essence of array processors is streamlining - breaking the operation into small steps, conducting simultaneous operations.

Synchronous, parallel, pipelined architecture is the mainstream design of array processors. The best representatives of the genre are 32-bit machines that perform full floating point calculations, though the Floating Point Systems, which accounts for 70% of the array processors' market, uses a 38-bit configuration as its standard. Most array processors are designed to optimize processing speed for the FFT, which is the most popular application using array processors. Some manufacturers choose to use fixed/block floating point processors to gain extra processing speed.

Most array processors are designed to work independently from the host once the program and data are down loaded, thus freeing the host for other purposes. Format conversion of data is generally implemented on the host interface board. ST-100 chooses to do the format conversion internally so as to maintain its 100M byte/sec input/output data transfer rate.

Most of the array processors come with a high level programming language which is compatible with Fortran 77. These high level languages incur certain inefficiency, perhaps a factor of two over assembly code. They usually include a library of pre-coded macros to facilitate process development and provide a macro assembly language for users desiring to code their own macros. Added to these, are simulation and debugger facilities for macro and process debugging, plus a host computer interface which allows transportation of the array processor from one host computer to another without reprogramming.

Different application software packages are available for each machine. Most of them include standard mathematics, signal processing, image processing, geophysical processing and simulation libraries.

STAR TECHNOLOGIES INC.

ST-100 is an array processor from Star Technologies Inc.. It uses synchronous, parallel, pipelined architecture. Its architecture employs multiple processors and a hierarchical high capacity data memory system. Bipolar VLSI circuits are used as the building blocks. The machine cycle is 40 ns; it can achieve up to 100 Mflops, which is the fastest comparable machine on the market and almost twice the speed of the second fastest machine in this survey.

Its multiple processors design results in a more general purpose computer code executing in a control processor, which consists of two Motorola 68000 microprocessors, and specialized microprocessor code which executes the special purpose processors. It has an MIMD (multiple instruction, multiple data) architecture. Its multilevel program structure and the hierarchical memory structure enable efficient usage of both host computer and array processor resources. They allow the array processor to perform a hierarchy of concurrent arithmetic and data movement operations independently without the need for host computer intervention.

Within the arithmetic processor, there are two add/subtract units, two multiply units and a divide/square root unit. The adders and multipliers are three-step pipelines, with each step executing at the clock rate of 40 ns. The divide/square root section is non-pipelined and requires 13 clock periods to compute. When the adders and multipliers are all in full operation, for example in convolution, the machine can operate at 100 Mflops.

To provide for very high memory bandwidth, two features of concurrency were introduced in ST-100 memory. The first feature is a very wide access path and the second is a highly interleaved memory. These give an aggregate data rate of up to 100M byte/second.

Another significant feature of ST-100 is that it can attach to and service multiple host computers concurrently. This is made possible by its production software which allows concurrent staging and execution of processes, enabling subsequent processors to be loaded and readied for execution while the current process executes. Its input/output subsystem is expandable to include independent input/output processors for each host system. Its maintenance software provides off-line and remote diagnostic capability.

The ST-100 software system consists of three major segments. The Development Software System is a set of Fortran programs residing in the host. It provides the ability to separately program all elements of the ST-100, allowing multiple levels of program optimization. These programs can be used to create application library modules. The simulator supports both the Fortran and micro code levels simulations while the debugger debugs micro codes only. The Production Software System couples the array processor to the host application program. It manages the allocation of array processor resources and directs requests from the user's application program to the control processor. It also schedules and controls requests from both multiple users and multiple hosts. In addition ST-100 provides a Maintenance Software System which includes a set of fault identification and isolation routines for diagnostic purposes.

The VAST software tool (the Vector and Array Syntax Translator) is designed to analyze DO loops in standard Fortran programs and convert those loops for which vectorization is possible into array operations. VAST creates a listing of the input program with diagnostic comments added to tell the user which loops are not vectorized and why. It also creates an enhanced version of the input program which includes ST-100 processes in place of the vectorized loops. Thus, the source remains transportable and readable. Conversion and debugging efforts are much reduced and made easier. The gap between the optimal efficiency of hand-coded operations, and total transportability and maintainability of standard Fortran on the host system is filled.

Star Technology has made large OEM sales with CDC and

GE. It appears to be a stable company.

Basic Configurations

Base Price : \$250,000 (512K word memory, host interface, development software and standard library)
Additional Memory : \$51,200/512K word
Additional Host Interface : \$10,000
Additional Host Software : \$12,500
Unix interface will be available four months after receiving any order.

Users' comments:

The Fortran modules provided are easy to program. However, setting up a module requires several hundred cycles, *therefore it is only worthwhile to call these modules if the vector is of several thousand elements.* That is, to fully utilize the machine, the problem to be solved must vectorize very well. As an alternative, one can use macro programming.

It is more tedious to program ST-100 as compare to other array processors. One has to keep track of how the data are arranged in the cache memory and there are two separate processors to be considered, the Arithmetic Control Processor and the Storage Move Processor. Typical speed is 25 Mflops if the problem being solved has only one DO loop. The bottle neck which one user experienced is at the interface between the cache memory and the arithmetic section. Since the cache memory only allow two reads and one write per cycle, in some applications, only one out of four of the arithmetic units can be running at each cycle. Also, if the host is a VAX, one may anticipate another bottle neck at the 1M byte interface between the host and the ST-100.

The manuals are good and extensive. They cover fine details and aim at audience with no previous knowledge on array processors. Software simulation package can be run on host machine for software development. One test program on simulation of equations of motion indicates the speed ratio among ST-100 : MARS-432 : FPS-120B is 14 : 57 : 96. The multi users facilities is only available at the later version and it works on one job at a time instead of time-sharing between jobs. The debugger is not available yet. STAR offers good services and responses quickly. They are very conscientious and helpful to problems encounter by users. The users are in general very pleased with the machine.

FLOATING POINT SYSTEMS

The FPS-5000 series is the most recent series of array processors introduced by the Floating Point Systems. It consists of the product groups - 5100, 5200, 5300 and 5400 with peak performance that ranges from 8 Mflops to 62 Mflops.

The family utilizes independent floating-point processing units, called Arithmetic Coprocessors. Data flow is simultaneously managed by a combination of independent Input/Output Processors and a central Control Processor. Arithmetic Coprocessors can be added as field-installable upgrades.

The internal structure of the Arithmetic Coprocessor is optimized for execution of the FFT, by virtue of the double memory access cycles. Special butterfly addressing hardware directs memory access during an FFT, in addition to simultaneous pipelined operations of the multiplier and two adders. The Arithmetic Coprocessor architecture includes internal memory, ALU, DMA communication and diagnostic elements. Using these parallel control and data path structures, 18 Mflops maximum performance can be achieved with an instruction cycle time of 167 ns. A maximum of three Arithmetic Coprocessors can be added to the FPS-5430 model. Adding the eight Mflops computation power from the Control Processor, a peak performance of 62 Mflops can be attained. However, data transfer between coprocessors must take place through the System Common Memory. Different data formats are used for the Arithmetic Coprocessor and the Control Processor. Thus efficiency and accuracy are reduced under some applications.

A Multiple Array Execution Language (MAXL) is provided for program development. It is a multiple processors control language that is a subset of Fortran 77 and it generates code for both Control Processor and the Arithmetic Coprocessor. AP-FORTRAN is another option that provides the capability of developing array processor routines using standard Fortran IV statements. AP-FORTRAN only executes on the Control Processor. Also available for program development are the assembler, simulator, linker, loader and debugger packages.

A programmable bit-slice interface processor (GPIOP) provides real time input/output applications. It can be programmed to interact with both the control requirements and the data transfer protocols of most peripheral devices.

The FPS-5000 series maintains software compatibility with previous FPS 38-bit processors, and is supported on a wide range

of host computers. Thus, the software support developed for FPS-100 and FPS-120B products is maintained and users are able to move existing applications on FPS-120B and FPS-100 onto the FPS-5000. The Floating Point Systems offers by far the most complete software libraries. This includes Standard and Advanced Mathematics, Signal, Image and Geophysical Processing and Simulation libraries.

The FPS-100 has been replaced by model FPS-5105. The FPS-120B is the most popular array processor on the market. It has been replaced by model FPS-5205.

Basic Configurations

Base Price : \$99,000 (4K word program memory,
(FPS-5430) 256K System Common Memory, host
interface and development
software)

Additional Program Memory : \$4,900/128K byte
Additional Data Memory : \$7,800/256K word
Software Library : ~\$1,000 each
Unix interface is available from outside source.
It is not support by FPS.

Users' comments:

Input/output is fast through the GPIOP but it is linearly addressing. The maximum memory is 1M word with page format which cannot be crossed. FPS-5205 is more cost effective than FPS-120B since it provides 256K memory at less than half the price. Cares have to be taken for the format conversion between host, control processor and arithmetic coprocessor (32-bit to 38-bit). One user complains that FPS-120B had a hardware problem where the back panels do not align properly and thus contacts are loose. Another user comments that the hardware is stable, but large amount of heat is produced and an air-conditioning room is a must. A user of the FPS-100 tried to split old program to run half on FPS-5205 control processor and half on the arithmetic processor, and he obtained twice the speed. FPS runs Fortran on the host and calls the array processor by subroutine calls and this causes a lot of overhead.

Extensive manual is provided. Software subroutines are useful and complete. The control processor runs on all subroutines from FPS-120B or FPS-100 model, but the arithmetic coprocessor only runs on a subset of the subroutines. One user said that FPS's debugger is useless and provide no information. The service from FPS is good and they response quickly to users' problems. In general, the users interviewed are happy with the machines.

NUMERIX CORPORATION

The MARS-432 array processor is the newest addition to the product line of Numerix Corporation. The key elements of its architecture are the Interface Processor(IP), the Data Processor(DP), and a 32-bit wide Data Bus(DBUS).

The IP controls the transfer of information on the 20M byte/sec DBUS. This bus is used to interface both programs and data to the DP. Data memory(DM) and Program Memory(PM) are included as part of the computational processor, the DP. This implementation feature allows additional speed to be obtained in arithmetic processing. If necessary, a data path between the two does exist, thereby allowing the DM to be used as a bulk storage area for large program memory.

The primary arithmetic elements are a multiplier and two ALUs. Again, this is optimized for the execution of the FFT. All units may execute in parallel and all are interconnected with multiple data paths. Each of these elements is commanded independently and may initiate an operation every 100 ns. This produces an arithmetic execution rate of 30 Mflops. Its throughput rates for vector add and vector multiply are higher than the FPS-5430 because of this shorter cycle time.

The Fortran Development System provides high-level language access to MAR-432. It consists of a Fortran compiler, linker and trace/monitor. The off-line development package includes the macroassembler, Loom and microcode debugger. The Loom is an utility that provides automatic microcode optimization. It automates the writing of pipelined code at the assembly language level and eliminates the need to hand code pipelined instructions. User interface to the MARS-432 at run time is taken care by AREX. AREX's transactions includes array processor initialization, input/output operation, and array function execution. MARS-432 offers application libraries for mathematic, signal processing, and geophysical processing.

Basic Configurations

Base Price : \$120,000 (1M word memory, host
interface, Fortran compiler,
fortran development system and
standard maths library)
Additional Memory : \$22,000/Mword
Extended Maths library : \$5,000 (includes image
processing routines)
Unix interface will be available if Numerix
gets an order for it.

Users' comments:

The Array Processor Run-time Executive (AREX) starts up slow, a long data stream has to be fed in to get a worthwhile run. The speed of the machine is ten times faster than FPS-120B when the whole program is downloaded into the machine. The machine has a large memory size and good integration of A/D and D/A interfaces but the input/output is slow. Its Fortran 77 compiler has some bugs and is not fully developed yet, but they can be get around.

MARS-432's manuals are primitive but total, new manuals are coming out every couple months. The software library is not as fully developed as FPS-120B but it is extremely useful. It is easy to program the machine. After service is good. Numerix can use modem to debug the machine without coming to the site. A good 3 weeks training course is offered to teach the users how to figure out the configuration according to the users' needs. One user got the machine for 6 months and the down time is more than the up time. The other two users interviewed are in general happy with the machine.

CSPI

The MAP-400 is the newest member in the MAP's family of 32-bit floating point array processors. The MAP's family uses a multiprocessor, three buses memory architecture to achieve high speed and high throughput. Its execution is divided among several specialized processors which operate in parallel. Though pipelined architecture is not implemented, its performance is comparable to machines of the same category and it better fits a certain kind of algorithm which is inefficient with pipelined architectures.

MAP-400 provides a choice of memory speeds (170 ns, 300 ns, 500 ns). Its capability can be added to an existing MAP-300 system. The hardware system contains two MAP-300 Arithmetic Processing Units (APU). MAP's multi-bus memory structure provides parallel data transfer for the APUs during calculation without interference. The data work buffers for the separate APUs may be configured by the user on separate memory buses thereby enabling both arithmetic units to run in parallel at full 12 Mflops.

The MAP-400 software system is an extension of the MAP 200/300 system, consisting of all operational, utility, and diagnostic software. The Operating System software consists of a set of programs which enables the user to call the supplied library subroutines. The calling programs, executed in the host, are

written in Fortran. In addition it includes several commands for executing a pair of function lists in parallel. The function lists may contain identical functions so that both APUs can perform the same task on two separate blocks of data at the same time. Because tasks are carried out in parallel, it is necessary to assure synchronization between processors. MAP uses a data driven hardware structure with queues internal to the processors to eliminate the need for concern with internal synchronization.

For those applications requiring direct analog input or output, MAP provides modules which handle such data in parallel with concurrent arithmetic. It also provides modules which allow direct access to and from disk and bulk memory storage device. Input and output buffers for the two units may reside on the same memory bus; however, the work buffers must be defined on separate buses to approach a doubling in processing speed.

Basic Configurations

Base Price	: \$54,850 (32K program memory)
Additional Data Memory	: \$19,500/256K byte
Additional Program Memory	: \$4,000/4K word
Hardware Interface	: \$3,500
Software System	: \$3,500 (Executive and Standard Library)

Mini-MAP is another fully programmable, 32-bit floating point array processor from CSPI. It interfaces with DEC computers. The basic system consists of 4 hex boards that plug directly into the Unibus.

One of the major overheads of array processing is passing data and commands between the host and the array processor. Mini-MAP's memory is designed to share memory with the host. Thus, the host CPU can process and move data in the Mini-MAP memory in the same way it uses its own Unibus memory. In effect, shared memory makes Mini-MAP a coprocessor.

The cycle of the machine is 125 ns. Each addition requires two cycles and each multiplication requires three cycles. It has a peak performance of seven Mflops and a benchmark testing of 7.8 ms for 1024 points complex FFT. It has no local program memory.

The Host Support Library is an extension to user's Fortran runtime library that is used for controlling Mini-MAP programs from a host Fortran program. Application program can be developed by MAP Control Language which is a Fortran subset language. A set of utility programs consisting of an assembler, compiler, linker, debugger and board-level diagnostics is also

provided.

Basic Configuration

Base Price : \$26,000 (4K byte memory)
Additional Memory : \$6,000/1M byte

ANALOGIC CORPORATION

Analogic's latest array processor is the AP-500. Its architecture combines a Motorola MC68000 based control processor with an internal 40-bit full floating point pipelined arithmetic logical unit. This provides better precision than the other 32-bit or 38-bit processors. Externally, it conforms with DEC 32-bit full floating point format.

The pipelined ALU consists of an Arithmetic Pipeline, an Address Generator and a Pipeline Sequencer. These three elements operate in parallel with the Control Processor and Input/Output, and fully utilize the AP Data memory bandwidth. Upon entry to the Arithmetic Pipeline, data are stored in the Pipeline Register Files (PRFs). Depending upon the algorithm, data can be moved between PRFs along the Bypass and Feedback Paths of the multiplier and adder. Thus, data may continually circulate within the Pipeline until final results are computed. This eliminates the need for time-consuming "Pipeline/Data Memory" data exchanges.

AP-500 offers modular subroutines instead of the more common Fortran compiler for the users. These subroutines are optimized to take full advantage of the AP-500's architecture. Applications can be programmed in Host or AP-500-resident High-Level or Assembly Languages. The AP Executive is available in single or multi-tasking versions. It manages internal AP-500 activities and resources to unburden and offload the host.

Basic Configuration

Base Price : \$35,000 (0.5M word memory, host
interface development software
and standard library)
: \$47,000 (1.0M word memory, host
interface development software
and standard library)
Assembler software : \$1,500
No Unix interface is available.

Users' comments:

It has a large memory of 0.5M word, twice that of standard FPS-5205. Data in memory can be accessed readily through the buffer and no paging is required. The user can create his own

buffer size. Input/Output speed is slow and is only one third that of FPS-5205. It takes up little physical space and consumes much less power than FPS-5205.

The software is immature, but it is being frequently updated. All the routines have to be preloaded before they are called. It is easy to program despite the fact that the high level language is not in Fortran and there is no Fortran Compiler. Service contract is recommended in the west coast since Analogic does not have much service personnel out there.

MERCURY COMPUTER SYSTEM

Mercury Computer Systems offers two array processors, the ZIP-3216 which performs 16-bit and 32-bit fixed point or block floating point operations, and the ZIP-3232 which will do 7 Mflops full floating point when available in first quarter 1985. The ZIP-3216 is the only machine in this survey which uses block floating point format. It is an interesting machine because of its cost and much of the vision works can use 16-bit computations.

The dual processors operate concurrently and are automatically synchronized in hardware. The AMD 29116 based Control Processor controls data flows. It delivers and extracts data of virtually any format to and from the Arithmetic Pipeline.

Within the ZIP architecture, the Arithmetic Pipeline, which is a single board, is independent of all other system components. It consists of a multiplier and a full ALU which provides a processing rate of 20 million computations per second in the 16-bit mode and 5 million computation per second in the 32-bit mode.

ZIP-3216 utilizes C-like instruction language, ZIP/C, which is augmented with special functions to control functional hardware. One of the characteristics of the language is the writing of arithmetic instructions. Output results can be directed to the FIFO and/or any of the accumulators. The compiler supports several product terms in the algebraic equation and the assignment of symbolic names to any variable. The ZIP/C compiler is able to distinguish between Control Processor and Arithmetic Pipeline codes contextually. The outputs are linked appropriately to create an application task file.

The ZIP Simulator/Debugger provides timing analysis of individual algorithm independently of the ZIP hardware. Its debugging facilities include a multiple viewing window into user-

defined elements of the ZIP. The user can display any component of ZIP. Multiple components can be simultaneously displayed and user-created display formats can be saved and recalled. The Simulator/Debugger allows ZIP programs to be single stepped, run until a user-specified condition is true, or run to completion. In addition trace files can be generated. To determine algorithm execution speed, the Simulation/Debugger generates timing statistics as well as utilization efficiencies for different components. The users can also simulate clock interrupts and Dual Direct Access Channel input/output data flow.

The algorithm library includes input/output and arithmetic functions for signal, image/graphic and scientific processing. The functions can be called within the user's host computer program or within the ZIP-3216 program.

Basic Configuration

Base Price : \$8,000 (128K byte program)
Additional Memory : \$3,000/512K byte
Software : \$2,000 (Development tools and
run time software)
Software Library : \$2,000-5,000 each
Unix interface is available

Users' comments:

The Multibus version was first delivered in July, and the users we talked to only have the machine for a few weeks. They do not have any hardware problem yet. The simulation package for program development is nice and easy to work with. The second and third drafts of the manuals are fairly complete. The manuals are getting to the point where they are usable. Some cosmetic and prettying up of the manuals are desirable.

MARINCO COMPUTER PRODUCT

The Marincos APB-3000 is a single board array processor. Its architecture allows a direct plug-in compatibility to the IEEE Multibus and is configurable to fit a 64k byte segment of memory. The APB-300 actually looks like a random access memory (RAM) board to the host processor and a high speed auxiliary BUS may be utilized for data transfer. It uses 24-bit full floating point and 16 bit integer arithmetic. This makes it less attractive than other machines.

The APB 300 machine uses a parallel 16-bit AMD29516 multiplier and a 16 bit microprocessor based AMD29116 ALU. It executes instructions in 125 ns and is capable of up to 8 million integer or 1 million floating point operations per second. Its

separated Program and Data Memories allows for parallel fetch and execution cycles. Utilizing two buses allows parallel operations such as inputting data in the ALU and loading the multiplier from the Data Memory. Its single board design actually cuts processing time by minimizing physical connection distance.

Data Memory is organized as 24-bit words internal to the machine. When 32-bit floating point data are moved from host memory, the lower 8 bits are dropped. In the reverse direction, 8 bits of zero are appended to the 24-bit memory word.

The APB-3000 is fully programmable and is available with an optional set of microcode development software. Marincos hierarchical assembler(HIASM) enables microprogramming in a high-level language. Marincos microassembler (MARASM) processes output from HIASM or direct user code, and produces an object code. Marincos APB monitor/debugger enables host based debugging of programs developed using HIASM or MARASM. It provides direct access to all APB-3000's registers, Data Memory, and Program Memory.

The operating microcode is available on flexible diskettes for use when the board is supplied with RAM for program memory. It can be supplied with PROMs containing the necessary microcode to perform operations such as digital filtering and FFT.

Basic Configurations

Base Price : \$4,250
Assembler : \$2,500

Users' comments:

The storage and retrieval of data require 1ms. This result is 100% overhead when doing floating point operations. The machine only supports addition and multiplication. The floating point format used is not of IEEE standard and its 24 bits representation causes reduction in precision. It has a small size program memory of 4K words. One of the user has troubles in reading the software tape and the board did not work properly when put on the Multibus on the SUN work station. Another user took one and a half month to figure out some hardware problem during first installation.

There is no software support, and standard subroutine like multiplication between matrix and vector has to be programmed by the user. Documentation is definitely not enough though readable. One user commented that he could only figure his way out by making some correct guesses from the manual. Marincos

has promised more documentation.s. Microcoding the machine is easy. Maringo provides some good service and deals with problems immediately.

SKY COMPUTER

Sky Micro Number Krunchers (SKYMNK) is a full 32-bit floating point array processor designed for use with 16-bit microcomputer systems. These array processors are easy to use as plug-in modules. SKYMNK is a pipelined, parallel coprocessor, which operates internally at speed of up to one Mflops. Its quoted speed is hard to achieve in reality since it has no local memory. A floating point operation must transfer two 32 bit operands, or four 16 bit words at about 4 microseconds minimum. Designed as a tightly-coupled coprocessor, the SKYMNK operates directly on data residing anywhere in host memory. Thus, separate Fortran calls which transfer data to and from the array processor are not necessary. Also, its DMA architecture eliminates the need for a costly additional memory. Overlap of DMA input/output with processing is automatic and user transparent while maximizing output.

Each SKYMNK is supplied with software support including inline driver, subroutine library, software simulator, and test diagnostic. The subroutine library contains routines for vector mathematic and signal processing. They can be called from either Fortran or Macro user program. The software simulator allows users to develop code for the SKYMNK without hardware actually installed.

Basic Configurations

Base Price : \$5,990
Software : \$1,200 (Simulator, diagnostic and documentation)
: \$4,000 (Assembly language source code)

CONCLUSION

ST-100 from Star Technologies is by far the fastest array processor in this study. The FPS-5000 series from the Floating Point Systems has the most software support because of its compatibility with previous models. The highest price/performance ratios are offered by ZIP 3216 from Mercury and FPS-5430 from the Floating Point Systems. It is likely that the introduction of ZIP-3232 will lower the price/performance ratio further.

Most of the machines use 32-bit floating point representation, however, few of them conform to IEEE floating point standard. Few of the manufacturers support the VAX-Unix system at present. However it is likely that most of them will provide full Unix support in the near future as Unix operating system is becoming more popular.

Though the maximum speed in Mflop is a good indication of the performance of a machine, we have to remember that for most algorithms this maximum speed cannot be reached. Some array processors try to boost their peak performance by providing more multiplier and adder units. This increase in speed is partly offset by the communication time required between computational units. Finally, the ease of programming is another important factor to be considered. This usually can only be learned through actual experience with the machines.

ACKNOWLEDGEMENTS

This work was supported by the Defense Advanced Research Projects Agency under Contract Number N00039-84-C-0211.

FURTHER INFORMATION

Star Technologies, Inc. (503-227-2052)
1200 Benjamin Franklin Plaza, One S.W. Columbia
Portland OR 97258.

Floating Point Systems, Inc. (503-641-3151)
P.O.Box 23489, Portland OR 97223.

Numerix Corp. (617-964-2500)
320 Needham Street, Newton MA 02161.

CSPI (617-272-6020)
40 Linnell Circle, Billerica MA 01821.

Analogic Corporation (617-246-0300)
Audubon Road, Wakefield MA 01880.

Mercury Computer System Inc. (617-458-3100)
Wannalancit Technology Center, 600 Suffolk Street
Lowell MA 01854.

Marinco Computer Product (619-453-5203)
11760 Sorrento Valley Road, San Diego CA 92121.

Sky Computer, Inc. (617-454-6200)
P.O.Box 8008, Lowell MA 01852.

Table of Comparison

	<u>SI100</u>	<u>FPS5430</u>	<u>MARS432</u>	<u>MAP400</u>	<u>AP500</u>	<u>ZIP3216</u>	<u>APB3000</u>	<u>SKY</u>	<u>FPS100</u>	<u>FPS120B</u>
Configurations										
Base Price(1) (dollar)	250000	99000	120000	55000	35000	8000	4250	5990	45000	55000
Max. Speed (Mflops)	100	62	30	24	9.4	5	1	1	8	12
Price/Speed Ratio	2500	1600	4000	2300	3700	1600	4250	5990	5600	4600
Float Point Format	Full	Full	Full	Full	Full	Block	Full	Full	Full	Full
Number of bits	32	38(2)	32	32	40	32	24	32	38(2)	38(2)
Integer Format	32	28(2)	32	32	32	32	16	-	28(2)	28(2)
Max. Program Memory (byte)	256k	128k	16k	224k	128k	10k	12k	host	32k	64k
Max. Data Memory (word)	8M	512k	512k	312k	912k	16M	16k	host	64k	512k
Hardware										
Machine cycle (ns)	40	167	100	200	125	200	125	143	250	167
I/O channel (Mbyte/sec)	100	12	20	36	25	20	20	4	8	12
Arithmetic Unit										
Adder/Subtract Unit	2	7	2	4	1	1	1	1	1	1
Multiplier Unit	2	4	1	4	1	1	1	1	1	1
Adder Cycle	3	5	5	1(3)	1(4)	1	-	-	2	2
Multiplier Cycle	3	5	5	2(3)	2(4)	1	-	-	3	3
Software										
High Level Language	FORTRAN	FORTRAN	FORTRAN	FORTRAN	Yes	C	FORTRAN	FORTRAN	FORTRAN	FORTRAN
Macro	Yes	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes	Yes
Assembler	Yes	Yes	-	Yes	-	Yes	-	Yes	Yes	Yes
Debugger (High Level)	-	-	Yes	-	-	Yes	-	-	Yes	Yes
Debugger (Low Level)	Yes	Yes	Yes	-	-	-	-	-	Yes	Yes
Linker	Yes	Yes	Yes	-	Yes	Yes	-	-	-	-
Library	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Math	-	Yes	Yes	-	-	Yes	-	-	Yes	Yes
Signal Proc.	-	Yes	Yes	-	-	Yes	-	-	Yes	Yes
Image Proc.	-	Yes	Yes	-	-	Yes	-	-	Yes	Yes
Geophysical	-	Yes	Yes	-	-	-	-	-	Yes	Yes
Simulation	-	Yes	Yes	-	-	-	-	-	Yes	Yes

Table of Comparison (Cont.)

	<u>ST100</u>	<u>FPS5430</u>	<u>MARS432</u>	<u>MAP400</u>	<u>AP500</u>	<u>ZIP3216</u>	<u>APB3000</u>	<u>SKY</u>	<u>FPS100</u>	<u>FPS120B</u>
Performance										
Vector Add (us)	0.04	0.25	0.2	0.3	0.49	0.4	-	1.1	-	-
Vector Multiply (us)	0.04	0.25	0.2	0.3	0.49	0.4	-	1.0	-	-
Vector Divide (us)	0.56	1.92	0.5	0.8	1.12	-	-	14.0	-	-
Complex 1024 pt. FFT (ms)	0.864	2.56	1.7	2.7	4.7	13.0	4.5	53.1	-	-
Complex 2D FFT 512X512 (ms)	400.0	500.0	-	-	-	4200	-	-	5100	3400
Matrix Multiply 100X100 (ms)	44.4	71.0	-	-	365.0	800.0	-	-	660	439
Convolution 128 by 8 (ms)	0.044	-	0.168	-	-	-	-	-	-	-
Convolution 1024 by 32(ms)	0.782	5.803	-	-	-	-	-	-	9.9	-
I/O Interface										
VAX-UNIX(5)	-	Yes	-	-	-	Yes	-	-	Yes	Yes
VAX-VMS(DEC)	Yes	Yes	Yes	Yes	Yes	Yes	-	-	Yes	Yes
PDP-11-RSX-11M	-	Yes	Yes	Yes	Yes	Yes	-	Yes	Yes	Yes
HP1000	-	Yes	-	Yes	Yes	-	-	-	Yes	Yes
Eclipse(DGC)	-	Yes	-	Yes	Yes	-	-	-	Yes	-
PE 3200	Yes	Yes	-	Yes	-	-	-	-	Yes	Yes
SEL(GOULD)	Yes	Yes	-	-	-	-	-	-	Yes	Yes
Harris	-	Yes	-	-	-	-	-	-	Yes	Yes
Prime	-	Yes	-	-	-	-	-	-	Yes	Yes
IBM4331,3081	Yes	Yes	-	-	-	-	-	-	Yes	-
Multibus	-	-	-	-	Yes	Yes	Yes	Yes	-	-
Unibus	Yes	Yes	Yes	Yes	Yes	-	-	Yes	Yes	-
Q-bus	-	-	-	-	Yes	Yes	-	Yes	-	-
Versabus	-	-	-	-	-	Yes	-	Yes	-	-
RS232	-	-	-	-	Yes	-	-	-	-	-
IBM-PC	-	-	-	-	-	Yes	Yes	-	-	-

Notes

- (1) detail configurations in the paper
- (2) FPS5XXX Arithmetic Coprocessor uses 32-bit floating point and 24-bit integer
- (3) one cycle is 240ns
- (4) one cycle is 160ns
- (5) full descriptions in the paper